AD-A062 915 CATHOLIC UNIV OF AMERICA WASHINGTON D C DEPT OF PHYSICS F/G 9/2
A FORTRAN CODE FOR THE CALCULATION OF SOUND PROPAGATION IN A RA--ETC(U)
JUN 78 A NAGL, G L ZARUR, H UEBERALL N00173-77-C-0008

UNCLASSIFIED NL

1 OF 2
AD
AD 62915

DEPARTMENT OF PHYSICS

The Catholic University of America
Washington, D.C. 20064

LEVEL II

(1)

A FORTRAN Code for the Calculation of Sound Propagation in a Range Dependent Ocean I. The Depth Functions.

Anton Nagl
Department of Physics
Catholic University of America
Washington, D.C. 20064

G.L. Zarur
Versar Inc.
Springfield, V.A. 22151

H. Überall
Department of Physics
Catholic University of America
Washington, D.C. 20064

Supported by Code 8120, Naval Research Laboratory, Washington, D.C. 20375 under Contract No. N00173-77-C-0008

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

(6) A FORTRAN CODE FOR THE CALCULATION OF SOUND PROPAGATION IN A RANGE DEPENDENT OCEAN, I. THE DEPTH FUNCTIONS.

Volume

5. TYPE OF REPORT & PERIOD COVERED
Technical Report(final)
Oct.15,1976-Jan.15,1977

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

(10) Anton Nagl, G.L. Zarur, and H. Überall

8. CONTRACT OR GRANT NUMBER(s)
NRL 173
N00173-77-C-0008 new

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Department of Physics
Catholic University of America
Washington DC 20064

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Program 62759N, Project SF52-552, Task SF52-552-691, Work Unit S01-94.802

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Sea Systems Command,
Washington DC 20362
(11) 7 Jun 78

12. REPORT DATE
June 7, 1978

13. NUMBER OF PAGES vi + 81 pp & 21 pp progr. listing

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
Code 8121 Naval Research Laboratory
Washington DC 20375
(12) 109 p.

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited

(9) Final rept. 15 Oct 76-15 Jan 77,

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

(16) F52552 (17) SF52552691

18. SUPPLEMENTARY NOTES

(15) N00173-77-C-0008

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Sound Propagation; Ocean Acoustics; Range Dependent Ocean; Depth functions; Arbitrary depth dependence; FORTRAN code; Normal Modes; Near-separation; Local depth functions; Eigenvalue problem

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a computer code which solves the wave equation for sound propagation in the ocean with arbitrary depth dependence and arbitrary but gradual range dependence of the sound velocity distribution.

The present volume (vol.1) is concerned with the depth functions which arise from the expansion of the velocity potential into a normal mode sum that leads to a near-separation of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

the range and depth dependent parts of the problem. The resulting depth functions satisfy locally a one-dimensional wave equation which together with the imposed boundary conditions defines an eigenvalue problem. The computer code described here permits to solve this problem for a wide variety of conditions. Deep ocean profiles can be treated for frequencies up to a few hundred Hz. For shallow water cases the frequency range easily extends to several kHz.

A further volume of this report will deal with the range functions and with the solution of the mode coupling problem.

ABSTRACT

This report describes a computer code which solves the wave equation for sound propagation in the ocean with arbitrary depth dependence and arbitrary but gradual range dependence of the sound velocity distribution.

The present volume (vol. 1) is concerned with the depth functions which arise from the expansion of the velocity potential into a normal mode sum that leads to a near-separation of the range and depth dependent parts of the problem. The resulting depth functions satisfy locally a one-dimensional wave equation which together with the imposed boundary conditions defines an eigenvalue problem. The computer code described here permits to solve this problem for a wide variety of conditions. Deep ocean profiles can be treated for frequencies up to a few hundred Hz. For shallow water cases the frequency range easily extends to several kHz.

A further volume of this report will deal with the range functions and with the solution of the mode coupling problem.

Acknowledgements

The support of the Naval Research Laboratory Code 8120 in the completion of this work is appreciated, as well as earlier support of the Office of Naval Research Code 486.

We wish to thank F. S. Chwieroth and F. Ingenito for helpful discussions, and R.H.Ferris for his interest.

CONTENTS

I. INTRODUCTION

The purpose of the computer code described in this volume is to find the solutions of the eigenvalue problem given by the differential equation

$$\frac{\partial^2 u_p(z,\vec{\rho})}{\partial z^2} + \left[\frac{\omega^2}{c^2(z,\vec{\rho})} - k_p^2(\vec{\rho})\right] u_p(z,\vec{\rho}) = 0 \qquad (1.1a)$$

and the boundary conditions

$$u_p(0,\vec{\rho}) = 0 \qquad (1.1b)$$

$$\lim_{z\to\infty} u_p(z,\vec{\rho}) = 0 \qquad (1.1c)$$

The variable $\vec{\rho}$ is considered a parameter. The solution consists of sets of eigenfunctions

$$u_p(z,\vec{\rho}) \qquad p = 1, 2, \ldots N(\vec{\rho})$$

and their associated eigenvalues $k_p(\vec{\rho})$ for each desired value $\vec{\rho}$.

Inputs are the sound frequency $f = \frac{\omega}{2\pi}$ and a function $c(z,\vec{\rho})$ corresponding in this case to the measured or simulated sound velocity distribution in some part of the ocean.

In the program it is assumed that $c(z,\vec{\rho})$ is given in the form of a two-dimensional grid of values:

$$c(z_1,\vec{\rho}_1),\ c(z_1,\vec{\rho}_2),\ \ldots\ c(z_1,\vec{\rho}_m)$$

$$c(z_k,\vec{\rho}_1),\ \ldots\qquad\qquad c(z_k,\vec{\rho}_m),$$

and that one can interpolate linearly between grid points. The set of values

$$c(z_i,\vec{\rho}_j) \qquad i = 1, \ldots k$$

will be referred to as the sound velocity profile at range point $\vec{\rho}_j$. In practice series of sound profiles are usually taken along a straight line course so the vectors $\vec{\rho}_j$ can be replaced by the scalar range variable ρ_j .

The problem as outlined above arises as part of the attempt of solving the wave equation

$$\rho(\vec{r})\vec{\nabla}\cdot\left[\rho^{-1}(\vec{r})\vec{\nabla}p(\vec{r})\right] + k^2(\vec{r})\,p(\vec{r}) = \delta(\vec{r}-\vec{r}_0)$$

describing the pressure filed of a unit point source (located at $\vec{r}_0$) in an inhomogeneous ocean medium of density $\vec{\rho}(\vec{r})$ and a propagation constant $k(\vec{r}) = \omega/c(\vec{r})$ where $c(\vec{r})$ is assumed to depend arbitrarily on the depth z and arbitrarily, but gradually, on the horizontal range coordinate $\vec{\rho} \equiv (x,y)$.

Assuming the density $\rho(\vec{r})$ to be constant throughout horizontal layers and defining a velocity potential $\phi(\vec{r})$ such that

$$p(\vec{r}) = \rho(\vec{r})\,\partial\phi(\vec{r})/\partial t, \tag{1.2}$$

one obtains for each layer a Helmholtz equation

$$\nabla^2\phi(\vec{r}) + k^2(\vec{r})\,\phi(\vec{r}) = \delta(\vec{r}-\vec{r}_0). \tag{1.3}$$

For a range independent sound velocity distribution, i.e. for $c(\vec{r}) = c(z)$ the solution of (1.3) can be written as a normal mode sum of the form

$$\phi(\vec{r}) = \sum_p \psi_p(\vec{\rho})\,u_p(z). \tag{1.4}$$

The "range functions" $\psi_p(\vec{\rho})$ then satisfy

$$\left[\nabla_\rho^2 + k_p^2(\vec{\rho})\right]\psi_p(\vec{\rho}) = \delta(\vec{\rho}-\vec{\rho}_0)\,u_p(z_0), \tag{1.5}$$

which is easily solved, while the "depth functions" $u_p(z)$ are obtained by solving (1.1) with $\vec{\rho} = const$. For arbitrary but gradual range dependence one can attempt a solution in the form of an almost separated

normal mode sum which takes the form

$$\phi(\vec{r}) = \sum_p \psi_p(\vec{\rho}) u_p(z, \vec{\rho}) \tag{1.5}$$

The equation satisfied by the range functions becomes now much more complicated involving terms which couple all the modes together, whereas the depth functions can still be obtained from (1.1) except that they are now no longer range independent, but gradually range-dependent "local" depth functions.

The problem of arbitrary z-dependence, and arbitrary but gradual range dependence of $c(\vec{r})$ is the case for which the programs described in this publication have been developed.

Vol. 1 is restricted to the discussion of the depth functions $u_p(z, \vec{\rho})$ and a method for calculating them. A discussion of the general range equation and a method for obtaining the modal range functions $\psi_p(\vec{\rho})$ together with a description of the associated computer code may be found in Vol. 2.

The method for solving (1.1) which was used in the computer code described here is based upon the assumption of piecewise linear sound velocity profiles. This assumption does not restrict the range of applicability of the code in any way. On the contrary, it enables one to obtain analytic solutions for the depth functions and thus makes it possible to obtain numerical solutions much more efficiently than by the method of numerical integration which is necessary for arbitrary profiles. In this way, normal mode theory becomes accessible to a much larger class of problems than otherwise possible. This is of particular

importance in view of the fact that the normal mode treatment is the most exact method of investigation sound propagation problems, which still works in cases where other methods (ray tracing, PE-method, etc.) fail. On the other hand, by inserting additional segments any velocity profile can, at least in principle, be approximated to any desired degree of accuracy by the present method. This would, of course, increase the computational effort again to unrealistic magnitudes. However, in practice, a very large number of segments is neither necessary nor is it in general possible, since for any given range point there is usually only a relatively small number of data points available to define the local sound velocity distribution.

II. DEPTH FUNCTIONS FOR A PIECEWISE LINEAR SOUND VELOCITY PROFILE

Since the boundary conditions are given at $z = 0$ and $z = \infty$, the depth functions $u(z, \vec{\rho})$ have to be found within these limits. The vertical sound velocity distribution at some particular range point $\vec{\rho}$ which determines the general solution of the depth function equation at this point is, as mentioned earlier, usually approximated by a piecewise linear function:

$$c(z) = c(z_i) + s_i^c (z - z_i) \qquad z_i \leq z \leq z_{i+1} \tag{2.1a}$$

with

$$s_i^c = [c(z_{i+1}) - c(z_i)]/(z_{i+1} - z_i) \qquad (i = 1, 2, \ldots K+1) \tag{2.1b}$$

The range $0 \leq z \leq \infty$ is thus subdivided into horizontal layers with boundaries at $z = z_i, z_{i+1}$ for the ith layer. The method to proceed with is therefore to find the solutions of the depth function equation within each layer and match the solutions at the layer boundaries. The conditions to be satisfied are that

$$\rho(z)\, u_\rho(z, \vec{\rho}) \quad \text{and} \quad \partial u_\rho(z, \vec{\rho})/\partial z \tag{2.2}$$

are continuous across each boundary. These follow from the requirement that the pressure

$$p = \rho \partial \phi / \partial t$$

and the velocity

$$\vec{v} = -\vec{\nabla} \phi$$

be continuous everywhere.

The water extends from $z = 0$ to $z = z_B$. The water density is assumed to be constant everywhere i.e.

$$\rho(z) = \rho_w \qquad 0 \leq z \leq z_B \tag{2.3}$$

The ocean floor is assumed to consist of isovelocity layers

with the density constant within each layer. At the present time only one layer is considered, hence the ocean floor is characterized by 2 numbers

$$\rho(z) = \rho_0 \qquad c(z) = c_B \qquad z > z_B .$$

However, if the need arises these assumptions can easily be generalized to several layers with linearly varying profiles.

Within the water $c(z,\vec{\rho})$ at some particular $\vec{\rho}$ varies between $c_{min}(\vec{\rho})$ and $c_{max}(\vec{\rho})$, where always $c_{max}(\vec{\rho}) \leq c_B(\vec{\rho})$.

A solution of the eigenvalue problem (1.1) is obtained only if the local vertical wave number $K_p(z,\vec{\rho})$,

$$K_p(z,\vec{\rho}) = \left\{ \left[\omega / c(z,\vec{\rho}) \right]^2 - k_p^2(\vec{\rho}) \right\}^{1/2} \tag{2.4}$$

is real for at least part of the range $0 < z < z_B(\vec{\rho})$. This limits the range of possible eigenvalues to

$$k_p(\vec{\rho}) < \omega / c_{min}(\vec{\rho}) . \tag{2.5a}$$

Discrete eigenvalues are obtained for

$$k_p(\vec{\rho}) > \omega / c_B(\vec{\rho}) . \tag{2.5b}$$

For

$$k_p(\vec{\rho}) < \omega / c_B(\vec{\rho}),$$

any value of k_p will provide a solution to eigenvalue problem (1.1). This report is concerned only with the calculation of discrete eigensolutions. With the assumptions made above the solution within the ocean floor, i.e. for the range

$$z_B(\vec{\rho}) \leq z < \infty$$

which satisfies the boundary conditons at ∞ is simply

$$\mu_p^B(z,\vec{\rho}) = a_p^B(\vec{\rho}) \exp\{ -\gamma_p(\vec{\rho}) z \} \tag{2.6a}$$

with

$$\gamma_p(\vec{\rho}) = \left\{ k_p^2 - \omega^2 / c_B^2(\vec{\rho}) \right\}^{1/2} . \tag{2.6b}$$

This leaves only the solutions for the range

$$0 \leq z \leq z_B(\vec{\rho}),$$

i.e. $\mu_p^w(z,\vec{\rho})$, the solutions within the water, to be determined.

They are obtained by solving (1.1a) with c(z) as given by (2.1) or more precisely with piecewise linear distribution of $\{\omega/c(z)\}^2$:

$$k_o^2(z) = \{\omega/c(z)\}^2 = \{\omega/c(z_i)\}^2 + s_i(z-z_i) \qquad z_i \leq z \leq z_{i+1} \qquad (2.7a)$$

with

$$s_i = \left\{\left[\omega/c(z_{i+1})\right]^2 - \left[\omega/c(z_i)\right]^2\right\}/(z_{i+1}-z_i). \qquad i=1,2,..k \qquad (2.7b)$$

Assuming a linear dependence of k_o^2 on z is compatible to first order with the assumption of a linear z-dependence of c if the variations in c(z) are small compared to the magnitude of c(z):

$$k_o^2(z) = \{\omega/c(z)\}^2 = \omega^2/\{c(z_i) + s_i^c(z-z_i)\}^2,$$

which with

$$\Delta c_i = |c(z_i+1) - c(z_i)| \ll c(z_i) \qquad (2.8)$$

can be written as

$$k_o^2(z) \cong \{\omega/c(z_i)\}^2 - 2\{\omega/c(z_i)\}^2\{s_i^c/c(z_i)\}(z-z_i)$$

Hence, if c(z) is a linear function with slope s_i^c then $k_o^2(z)$ is also approximately a linear function with slope $s_i = -2\omega^2 s_i^c/\{c(z_i)\}^3$ and vice versa, if (2.8) holds. Condition (2.8) is satisfied for sound velocity profiles since typically

$$\Delta c < 50\ m/sec, \qquad c \cong 1500\ m/sec.$$

Proceeding with the assumption that $k_o^2(z)$ can be linearized according to (2.8) the depth function equation for the ith layer

at some particular $\vec{\rho}$ becomes

$$d^2u_p(z,\vec{\rho})/dz^2 + \left\{\left[\omega/c(z_i,\vec{\rho})\right]^2 + s_i(\vec{\rho})(z-z_i) - k_p^2(\vec{\rho})\right\} u_p(z,\vec{\rho}) = 0$$

or

$$d^2u_p(z,\vec{\rho})/dz^2 + \left\{d_{pi}(\vec{\rho}) + s_i(\vec{\rho})z\right\} u_p(z,\vec{\rho}) = 0$$

with

$$d_{pi}(\vec{\rho}) = \left\{\omega/c(z_i,\vec{\rho})\right\}^2 - s_i(\vec{\rho})z_i - k_p^2(\vec{\rho}) . \qquad (2.9)$$

The substitution

$$\zeta_{pi}(z,\vec{\rho}) = -\left\{\left[s_i(\vec{\rho})\right]^2\right\}^{-1/3}\left\{d_{pi}(\vec{\rho}) + s_i(\vec{\rho})z\right\} \equiv \alpha_i(\vec{\rho})\left[z+\beta_i(\vec{\rho})\right] \qquad (2.10)$$

transforms this equation into the Airy equation

$$d^2u_p/d\zeta^2 = \zeta u_p .$$

Therefore, the solution for the pth mode in the ith layer in the water becomes

$$u_{pi}^{w}(z,\vec{\rho}) = a_{pi}Ai(\zeta_{pi}) + b_{pi}B_i(\zeta_{pi}) . \qquad z_i \le z \le z_{i+1} \qquad (2.11)$$

There are 2 constants to be determined for each layer. By watching the solutions and their derivatives at the layer boundaries the coefficients of one layer can be expressed in terms of quantities from the preceding layer.

Abbreviating $\begin{cases} u_{pi}(z) = u_{pi}^{w}(z,\vec{\rho}), \quad u_{pi}'(z) = u_{pi}^{w\,\prime}(z), \\ A_i = Ai(\zeta_{pi})\big|_{z=z_i}, \quad A_{i-1} = Ai(\zeta_{p\,i-1})\big|_{z=z_i}, \quad \alpha_i = d\zeta_i/dz, \end{cases}$

one can write the matching conditions at the boundary between the ith and the (i-1)st layer as

$$u_{p\,i-1}(z_i) \equiv a_{p\,i-1}A_{i-1} + b_{p\,i-1}B_{i-1} = a_{pi}A_i + b_{pi}B_i \equiv u_{pi}(z_i) \qquad (2.12)$$

$$u'_{p\,i-1}(z_i) \equiv -\alpha_{i-1}\left[a_{p\,i-1}A'_{i-1} + b_{p\,i-1}B'_{i-1}\right] = -\alpha_i\left[a_{pi}A_i' + b_{pi}B_i'\right] \equiv u_{pi}'(z_i).$$

Solving these two equations for a_i and b_i in terms of $u_{i-1}(z_i)$

and $u'_{i-1}(z_i)$ yields

$$a_{pi} = \pi \left\{ u_{pi-1}(z_i) B_i' + \alpha_i^{-1} u'_{pi-1}(z_i) B_i \right\} \qquad (2.13)$$

$$b_{pi} = \pi \left\{ -u_{pi-1}(z_i) A_i' - \alpha_i^{-1} u'_{pi-1}(z_i) B_i \right\}.$$

The first derivatives of the depth functions are obtained from (2.11) as

$$-\alpha_i^{-1} u_{pi}^{w\,\prime}(z,\vec{\rho}) = a_{pi} A_i'(\zeta_{pi}) + b_{pi} B_i'(\zeta_{pi}), \qquad (2.14)$$

keeping in mind that the derivatives of the Airy functions are with respect to ζ. When matching derivatives at the layer boundaries one has of course to use derivatives with respect to z, as was done in (2.12). However, in order to propagate the solutions for each mode from one layer to the next, one can, as suggested by eqs. (2.13) and (2.14), use just as well ζ-derivatives. This approach was taken in the program described here. Defining

$$u'_{pi}(z_i)_\zeta = -\alpha_i^{-1} u'_{pi}(z_i),$$

the coefficients (2.13) then become:

$$\begin{aligned} a_{pi} &= \pi \left[u_{pi-1}(z_i) B_i' - T\, u'_{pi-1}(z_i)_\zeta B_i \right] \\ b_{pi} &= \pi \left[-u_{pi-1}(z_i) A_i' + T\, u'_{pi-1}(z_i)_\zeta A_i \right] \end{aligned} \qquad (2.13')$$

with $T = \alpha_{i-1}/\alpha_i$.

An alternate form for propagating the solutions, also used in the program, is

$$\begin{aligned} u_{pi}^{w}(z,\vec{\rho}) &= \tau_{11}\, u_{pi-1}(z_i) + \tau_{12}\, u'_{pi-1}(z_i)_\zeta \\ u_{pi}^{w\,\prime}(z,\rho)_\zeta &= \tau_{21}\, u_{pi-1}(z_i) + \tau_{22}\, u'_{pi-1}(z_i)_\zeta \end{aligned} \qquad (2.15)$$

where

$$\tau_{11} = \pi \{ Ai'(\zeta_{pi}) B_i - Bi(\zeta_{pi}) Ai' \}$$
$$\tau_{12} = \pi T \{ Bi(\zeta_{pi}) A_i - Ai(\zeta_{pi}) B_i \}$$
$$\tau_{21} = \pi \{ Ai'(\zeta_{pi}) B_i' - Bi(\zeta_{pi}) Ai' \}$$
$$\tau_{22} = \pi T \{ Bi'(\zeta_{pi}) A_i - Ai'(\zeta_{pi}) B_i \} .$$

Using eqs. (2.11), (2.13) and (2.14), the full solution $u^w(z, \vec{\rho})$ in the water can be determined in terms of just two unknown constants, which can be e.g. a_{p1} and b_{p1}, i.e. the coefficients in the layer nearest the surface. One constraint for these coefficients comes from the boundary conditions at the surface:

$$u_p^w(0, \vec{\rho}) = 0 \tag{2.16}$$

Hence from (2.11):

$$a_{p1} A_1 + b_{p1} B_1 = 0 . \tag{2.17}$$

If one chooses for a_{p1} the arbitrary but convenient value

$$a_{p1} = B_1 , \tag{2.18}$$

then b_{p1} is determined:

$$b_{p1} = -A_1 , \tag{2.19}$$

and one obtains for the slope of the solution just below the surface from (2.14)

$$u_{p1}^{w\,\prime}(0, \vec{\rho}) = -\alpha_1 (B_1 A_1' - A_1 B_1') = \alpha_1(\vec{\rho})/\pi . \tag{2.20}$$

This choice of coefficients ensures that all solutions start out with a positive derivative near the surface:

$$u_{p1}^{w\,\prime}(0, \vec{\rho}) > 0 , \tag{2.21}$$

which makes the ζ derivative

$$u_{p1}^{w\,\prime}(0, \vec{\rho})_\zeta = -[\alpha_1(\vec{\rho})]^{-1} u_{p1}^{w\,\prime}(0, \vec{\rho}) = -\pi^{-1} \operatorname{sgn}[\alpha_1(\vec{\rho})] . \tag{2.22}$$

Using (2.11) and (2.14) with (2.18) and (2.19) one can determine the solution and the derivative at the lower boundary of the first layer. In terms of these quantities the coefficients for the second layer can be found using (2.13). In this way the solution $u^W(z, \vec{\rho})$ can be propagated to $z = z_B$, the ocean floor. There the functions $u_p^W(z, \vec{\rho})$ and $u_p^B(z, \vec{\rho})$ have to be matched. This leads to the conditions

$$\rho_W u_p^W(z_B, \vec{\rho}) = \rho_B u_p^B(z_B, \vec{\rho}) \tag{2.23}$$

$$\left.\frac{\partial u_p^W(z,\vec{\rho})}{\partial z}\right|_{z=z_B} = \left.\frac{\partial u_p^B(z,\vec{\rho})}{\partial z}\right|_{z=z_B} = -\gamma_p(\vec{\rho})\, u_p^B(z_B, \vec{\rho}). \tag{2.24}$$

As indicated in (2-7a), u_p^B contains one undetermined coefficient. It can be found by using (2.23). The choice (2.18) was made because it provides a convenient starting point for numerical calculations, as will become apparent below. But it was arbitrary in the sense that solutions calculated with this choice do in general not satisfy the normalization condition for normal mode solutions, in this case:

$$\int_0^\infty \frac{\rho(z)}{\rho_W} |u_p(z, \vec{\rho})|^2 \, dz = 1 \quad . \tag{2.25}$$

To remedy this situation the integral over the square of the unnormalized solution is evaluated to find the necessary normalization factor

$$A_p = \left\{ \int_0^{z_B} |u_p^W(z,\vec{\rho})|^2 dz + (\rho_B/\rho_W) \int_{z_B}^{\infty} |u_p^B(z,\vec{\rho})|^2 dz \right\}^{-1/2} \tag{2.26}$$

where

$$\int_0^{z_B} |u_p^W(z,\vec{\rho})|^2 dz = \sum_{i=1}^{N} \frac{1}{\alpha_{pi}(\vec{\rho})} \int_{\zeta_{pi}(z_{i-1},\vec{\rho})}^{\zeta_{pi}(z_i,\vec{\rho})} \left[a_{pi} A_i(\zeta_{pi}) + b_{pi} B_i(\zeta_{pi}) \right]^2 d\zeta_{pi} \tag{2.27a}$$

$$\frac{\rho_B}{\rho_W} \int_{z_B}^{\infty} |u_p^B(z,\vec{\rho})|^2 dz =$$

$$= \frac{\rho_B}{\rho_W} \int_{z_B}^{\infty} \left[a_{pN}^B(\vec{\rho}) \right]^2 e^{-2\gamma_p(\vec{\rho}) z} = \frac{\rho_B}{\rho_W} \frac{\left[a_{pN}^B(\vec{\rho}) \right]^2}{2\gamma_p(\vec{\rho})} e^{-2\gamma_p(\vec{\rho}) z_B} . \tag{2.27b}$$

The first integral can also be carried out analytically using

$$\int_{\zeta_1}^{\zeta_2} Ai(\zeta)^2 d\zeta = \left[\zeta (Ai(\zeta))^2 - (Ai'(\zeta))^2 \right]_{\zeta_1}^{\zeta_2} . \tag{2.28a}$$

Similarly

$$\int_{\zeta_1}^{\zeta_2} Ai(\zeta) Bi(\zeta) d\zeta = \left[\zeta Ai(\zeta) Bi(\zeta) - Ai'(\zeta) Bi'(\zeta) \right]_{\zeta_1}^{\zeta_2} ,$$

and

$$\int_{\zeta_1}^{\zeta_2} (Bi(\zeta))^2 d\zeta = \left[\zeta (Bi(\zeta))^2 - (Bi'(\zeta))^2 \right]_{\zeta_1}^{\zeta_2} . \tag{2.28b}$$

(2.28c)

These expressions follow from

$$\int C_1(\zeta)\, C_2(\zeta)\, d\zeta = \zeta\, C_1(\zeta)\, C_2(\zeta) - C_1'(\zeta)\, C_2'(\zeta)$$

by differentiating with respect to ζ:

$$C_1 C_2 = C_1 C_2 + \zeta\,(C_1' C_2 + C_1 C_2') - C_1'' C_2' - C_1' C_2'' .$$

Upon rewriting this equation as

$$C_2'\,(C_1'' - \zeta C_1) + C_1'\,(C_2'' - \zeta C_2) = 0$$

it is apparent that it is satisfied for C_1 and C_2 being any solution of the Airy equation.

The functions $u_p(z, \vec{\rho})$ as discussed so far still do not satisfy eq. (2.24). The coeffic nts of the two partial solutions $u_p^W(z, \vec{\rho})$ and $u_p^B(z, \vec{\rho})$ were chosen such that the boundary conditions at $z = 0$ and $z = \infty$ were satisfied and that $u_p(z, \vec{\rho})$ is continuous everywhere. The derivative $u_p'(z, \vec{\rho})$ is at this point continuous everywhere inside the water and inside the ocean floor. However, so far no provisions have been made to make it also continuous at $z = z_B$. In order to achieve this, u_p^W and u_p^B have to be adjusted such that (2.24) is satisfied. This can be done by choosing the proper values for $k_p(\vec{\rho})$ which enter u_p^W and u_p^B through (2.9) and (2.6b). In this sense eq. (2.24) provides the characteristic equation for the eigenvalue problem at hand. A more convenient form of the characteristic equation

is found by dividing (2.24) by (2.23), which corresponds to matching logarithmic derivatives at the ocean floor:

$$\left.\frac{\partial u_p^W(z,\vec{\rho})}{\partial z}\right|_{z=z_B} = -\frac{\rho_W}{\rho_B}\gamma_p(\vec{\rho})\,u_p^W(z_B,\vec{\rho}). \tag{2.29}$$

The characteristic equation defining the eigenvalues $k_p(\vec{\rho})$ can, of course, be established at the boundary between any two layers. It turns out that sometimes it is advantageous to propagate the solution from infinity up to the boundary between two water layers. The characteristic equation is then

$$u_{p\downarrow}(z_M,\vec{\rho})\,u'_{p\uparrow}(z_M,\vec{\rho}) = u'_{p\downarrow}(z_M,\vec{\rho})\,u_{p\uparrow}(z_M,\vec{\rho}), \tag{2.30}$$

assuming the match occurs at the boundary between the (M-1)st and the Mth layer. Here $u_{p\downarrow}$ is the solution started at $z = 0$, $u_{p\uparrow}$ the solution started at ∞ .

Both (2.23) and (2.30) are used in the paper and will be referred to later on.

III. ARGUMENTS OF THE AIRY FUNCTIONS AND GENERAL PROPERTIES OF THE EIGENVALUE SPECTRUM

Since the Airy functions play a central role in the problem discussed in this report it is useful to point out some general properties of these functions and to try to get an idea of how the various parameters which enter the problem affect the regions in which the Airy functions are needed. It turns out that the arguments of the Airy functions within a particular layer can be expressed to a good approximation in a relatively simple way in terms of the frequency, the slope of the velocity profile and a quantity with the dimension of a sound velocity which indicates how far the eigenvalue is above or below the wave number corresponding to the local sound speed. This formula can not only be used to determine the range over which the arguments of the Airy functions vary for a particular profile but also to estimate the total number of modes possible and the average separation between successive eigenvalues as a function of mode number.

The Airy functions are characterized by the observation that they behave like an oscillating function for negative arguments and like an exponentially rising or falling function for positive arguments. The characteristics are clearly brought out by the asymptotic expressions for large arguments ($|\zeta| \gg 1$):

$$Ai(\zeta) \sim \frac{e^{-\eta}}{2\sqrt{\pi}\,\zeta^{1/4}} \left\{ 1 - \frac{15}{216}\frac{1}{\eta} + \cdots \right\} \qquad \left(\eta = \tfrac{2}{3}\zeta^{3/2}\right)$$

$$Bi(\zeta) \sim \frac{e^{\eta}}{\sqrt{\pi}\,\zeta^{1/4}} \left\{ 1 + \frac{15}{216}\frac{1}{\eta} + \cdots \right\}$$

$$Ai(-\zeta) \sim \frac{1}{\sqrt{\pi}\,\zeta^{1/4}} \left\{ \left(1 - \frac{35\times 99}{2\times 216^2}\,\frac{1}{\eta^2} + \cdots\right) \sin\left(\eta + \frac{\pi}{4}\right) - \left(\frac{15}{216}\,\frac{1}{\eta} - \cdots\right) \cos\left(\eta + \frac{\pi}{4}\right) \right\} \qquad (3.1)$$

$$Bi(-\zeta) \sim \frac{1}{\sqrt{\pi}\,\zeta^{1/4}} \left\{ \left(1 - \frac{35\times 99}{2\times 216^2}\,\frac{1}{\eta^2} + \cdots\right) \cos\left(\eta + \frac{\pi}{4}\right) + \left(\frac{15}{216}\,\frac{1}{\eta} - \cdots\right) \sin\left(\eta + \frac{\pi}{4}\right) \right\} .$$

The argument of the Airy functions is given by (2.10) with $d_{pi}(\vec{\rho})$ and $s_i(\vec{\rho})$ defined by (2.9) and (2.7b). It follows that (ignoring the $\vec{\rho}$ -dependence and writing $c(z_i)$ as c_i etc.):

$$\zeta_{pi}(z) = -(s_i)^{-2/3}\left[k_o^2(z) - k_p^2\right] \qquad (3.2)$$

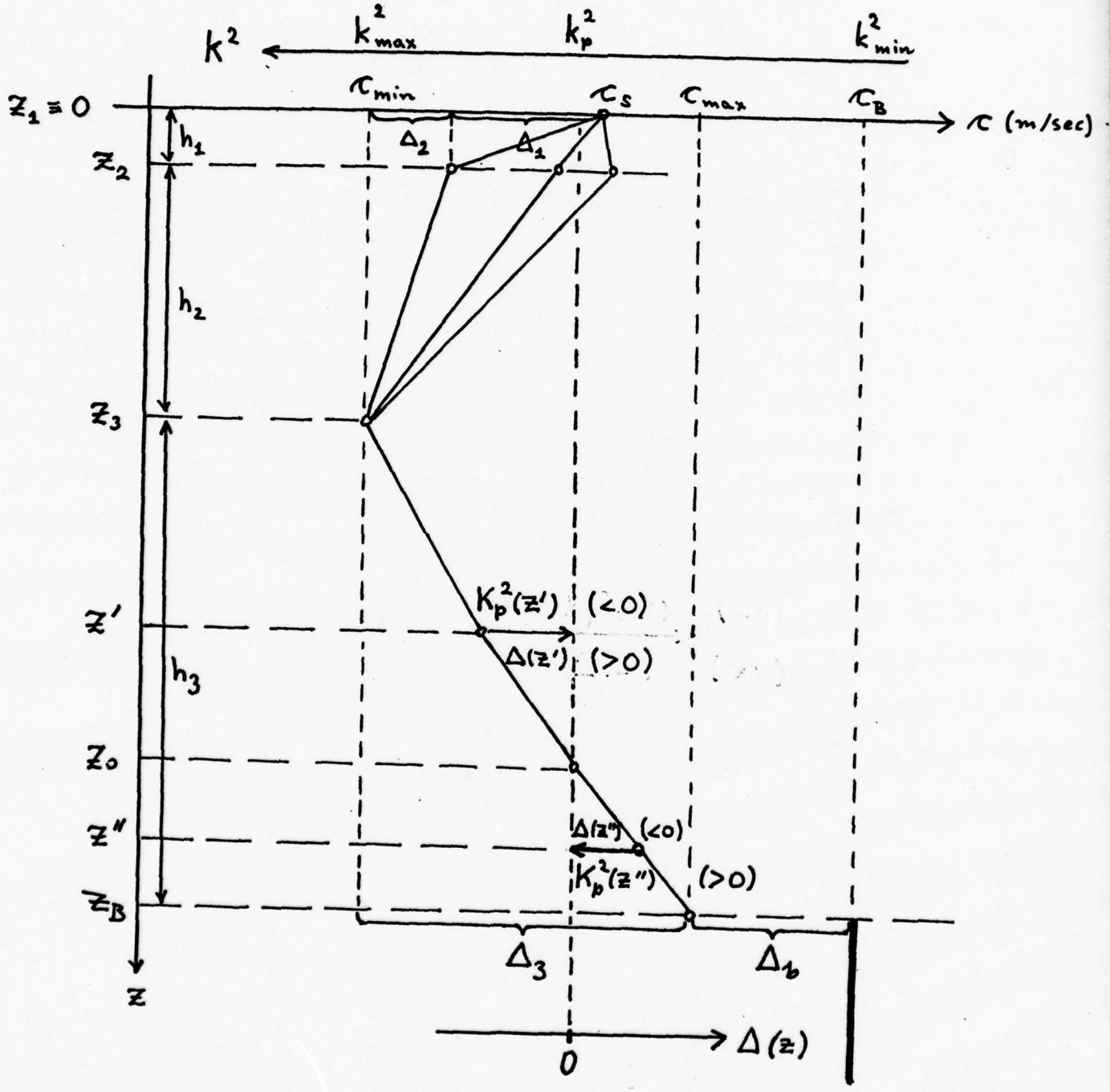
$$k_o^2(z) = \frac{\omega^2}{c_i^2} + s_i(z - z_i)$$

denoting the k^2 - value as a function of z along the profile. As indicated by (2.5a) and (2.5b) the range of discrete eigenvalues is restricted to

$$\frac{\omega}{c_B} = k_{min} < k_p < k_{max} = \frac{\omega}{c_{min}} .$$

This is indicated in Fig. 1 where schematically a typical profile is shown.

Fig. 1


k^2
k^2_{max}
k^2_p
k^2_{min}
$z_1 \equiv 0$
τ_{min}
τ_s
τ_{max}
τ_B
τ (m/sec)
h_1
Δ_2
Δ_1
z_2
h_2
z_3
$K_p^2(z')$ (<0)
z'
$\Delta(z')$ (>0)
h_3
z_0
z''
$\Delta(z'')$ (<0)
$K_p^2(z'')$
(>0)
z_B
Δ_3
Δ_b
z
0
$\Delta(z)$

The quantity which essentially determines the argument of the Airy functions as a function of z for a particular mode p is the local vertical wave number $K_p(z)$ defined by

$$K_p^2(z) = k_o^2(z) - k_p^2 \quad . \tag{3.3}$$

Defining new variables

$$\mathcal{K}_o^2(z) = -k_o^2(z) + \omega^2/c_{min}^2$$

$$\mathcal{K}_p^2 = -k_p^2 + \omega^2/c_{min}^2$$

with the ranges

$$0 \leq \mathcal{K}_o^2(z) \leq \frac{\omega^2}{c_{min}^2} - \frac{\omega^2}{c_{max}^2} = \frac{\omega^2}{c_{min}^2} - \frac{\omega^2}{(c_{min}+\Delta_3)^2} \cong \frac{2\omega^2}{c_{min}^3}\,\Delta_3$$

and

$$\mathcal{K}_p^2 = \begin{cases} 0 & (p=1) \\ \dfrac{\omega^2}{c_{min}^2} - \dfrac{\omega^2}{c_B^2} = \dfrac{\omega^2}{c_{min}^2} - \dfrac{\omega^2}{(c_{min}+\Delta_3+\Delta_b)^2} \cong \dfrac{2\omega^2(\Delta_3+\Delta_b)}{c_{min}^3} & (p=N) \end{cases}$$

one can see that

$$K_p^2(z) = \mathcal{K}_p^2 - \mathcal{K}_o^2(z)$$

varies between

$$-2\omega^2\Delta_3/c_{min}^3$$

for p = 1 (i.e. for the eigenvalue with the lowest mode number) and at $z = z_B$ (the z-value with the highest sound velocity), and

$$2\omega^2 (\Delta_3 + \Delta_b)/c_{min}^3$$

for p = N (for the eigenvalue with the highest mode number and at $z = z_3$ (the depth with the minimum sound speed), with values

$$2\omega^2 \Delta(z)/c_{min}^3$$

in between, where

$$-\Delta_3 \leq \Delta(z) \leq \Delta_3 + \Delta_b . \tag{3.4}$$

Hence the coordinate axis of $\Delta(z)$ has the same units as c(z) but its origin is not fixed in Fig. 1. Instead, it shifts around and its position corresponds to the location of the square of the eigenvalue (k_p^2) on the k^2 - axis.

The slopes s_i can be written as

$$\begin{aligned} s_i &= \frac{1}{h_i}\left(\frac{\omega^2}{c_{i+1}^2} - \frac{\omega^2}{c_i^2}\right) \\ &\cong 2 s_i^c \omega^2/\omega_{min}^2 = 2(\Delta i/h_i)(\omega^2/c_{min}^3). \end{aligned} \tag{3.5}$$

With these definitions and approximations the argument of the Airy functions in the ith layer for the pth mode can be written as

$$\zeta_{pi}(z) = -\frac{2^{1/3}}{c_{min}} \left(\frac{\omega}{\Delta_i/h_i}\right)^{2/3} \Delta(z). \tag{3.6}$$

Hence, it turns out that positive arguments are generally obtained for low wave numbers, negative arguments for high wave numbers. The slope of the profile in a particular layer does not affect the sign of the argument. However, the absolute value of the argument may grow very large as the slope approaches zero. Layers with small or vanishing vertical sound speed gradients are frequently encountered near the surface and of course near sound speed minima. Some care is necessary to maintain the accuracy of the solutions in these cases as will be shown below.

In order to get some numerical values the following parameters were chosen:

$$f = 250\,\text{Hz}$$

$$s_1^c = \Delta_1/h_1 = 0.1,\ 5,\ 100\ \text{m/sec per 100 m}$$

$$s_2^c = \Delta_2/h_2 = 5\ \text{m/sec per 100 m}$$

$$s_3^c = \Delta_3/h_3 = 1.5\ \text{m/sec per 100 m}$$

$$c_{min} \cong 1500\ \text{m/sec.}$$

The three values chosen for s_1^c correspond to low, average and high sound velocity gradients as usually encountered at small or intermediate depths. The value chosen for s_3^c is typical for greater depths. With these numbers are obtained for the Airy function arguments:

$$\zeta_{p1}(z) = -11.4\,\Delta(z),\ -0.84\,\Delta(z),\ -0.114\,\Delta(z)$$

$$\zeta_{p2}(z) = -0.84\,\Delta(z)$$

$$\zeta_{p3}(z) = -1.87\,\Delta(z).$$

For f = 25 Hz all coefficients are smaller by a factor of $10^{2/3} = 4.64$. For the lowest eigenvalue:

$$\begin{aligned} \Delta(0) &\cong -(\Delta_1 + \Delta_2) \\ \Delta(z_2) &\cong -\Delta_2 \\ \Delta(z_3) &\cong 0 \\ \Delta(z_B) &\cong -\Delta_3 \,. \end{aligned} \tag{3.7a}$$

For the highest eigenvalue:

$$\begin{aligned} \Delta(0) &\cong \Delta_3 + \Delta_b - \Delta_1 - \Delta_2 \\ \Delta(z_2) &\cong \Delta_3 + \Delta_b - \Delta_2 \\ \Delta(z_3) &\cong \Delta_3 + \Delta_b \\ \Delta(z_B) &\cong \Delta_b \,. \end{aligned} \tag{3.7b}$$

The parameters encountered in a deep ocean case may be e.g.

$$h_1 = 400\,m, \quad h_2 = 600\,m, \quad h_3 = 3000\,m$$

$$\Delta_1 = 0.4\,m/sec, \quad \Delta_2 = 30\,m/sec, \quad \Delta_3 = 45\,m/sec, \quad \Delta_b = 300\,m/sec. \tag{3.8}$$

This leads to the following Airy function arguments for the lowest mode:

$\sim$ 346 at the surface (z = 0)

$\sim$ 340 at the end of the first layer ($z = z_2$)

$\sim$ 25 at the beginning of the second layer ($z = z_2$)

$\sim$ 0 at the end of the second layer ($z = z_3$)

$\sim$ 0 at the beginning of the third layer ($z = z_3$)

$\sim$ 84 at the ocean floor ($z = z_B$)

The corresponding numbers for the highest mode are

$\sim$ -3570, -3575, -263, -288, -644, -560 for the quoted value of Δ_b

$\sim$ -165, -170, -4, -38, -84, 0 for $\Delta_b = 0$.

It is apparent that a combination of high frequency, small slopes in the profile, very low and very high mode numbers can result in very large positive or negative arguments for the Airy functions. From the asymptotic expressions for the Airy functions one can see that large positive arguments are particularly serious since they can lead to exponents of the order of several thousand. For the lowest mode e.g., the Airy functions are of the order of unity near $z = z_2$, but in the example given, of the order

$$e^{2/3\ 84^{3/2}} \cong 10^{223}$$

near $z = z_b$, i.e. the eigenfunctions for the lowest mode grow over more than 200 orders of magnitude between the ocean floor and the location of the sound speed minimum. For very small slopes which may occur near the surface or near a sound speed minimum the exponents may be even higher. As the small slopes

do not usually persist over large distances the growths of the Airy functions between the beginning and the end of a layer are relatively small, but the coefficients obtained when matching the solutions to the other layers become small to the same order as the Airy function grows large, and near-cancellation of very large numbers occurs. Special precautions have to be taken to avoid overflows and underflows in the computer and to ensure that the accuracy of the results does not deteriorate due to the near cancellation of large numbers. This problem will be addressed in the next section.

Eq. (3.6) can be used to estimate the number of discrete modes available for a particular profile and frequency.

The highest mode is an oscillatory function all the way from the surface to the ocean floor since the arguments of the Airy functions are negative throughout. Also since the arguments are large in most of the region the Airy functions can be approximated by

$$Ai(-\zeta) \propto \sin\left(\frac{2}{3}\zeta^{3/2} + \frac{\pi}{4}\right)$$

$$Bi(-\zeta) \propto \sin\left(\frac{2}{3}\zeta^{3/2} + \frac{3\pi}{4}\right).$$

By keeping track of the phase differences of the sines between the two boundaries of each layer and adding them up over the whole water depth one can approximately determine the number

of half-waves accomodated in the water column and thus deduce the wave number for the highest mode and at the same time the total number of modes supported by the profile:

$$N \cong \frac{2}{3\pi} \sum_{j=1}^{M} \left| \left[-\zeta_{N_j}(z_{j+1}) \right]^{3/2} - \left[-\zeta_{N_j}(z_j) \right]^{3/2} \right| , \quad (3.9)$$

or using (3.6)

$$N \cong \frac{2}{3\pi} \frac{\sqrt{2}\,\omega}{c_{min}^{3/2}} \sum_{j=1}^{M} \frac{h_j}{\Delta_j} \left| \Delta^{3/2}(z_{j+1}) - \Delta^{3/2}(z_j) \right| . \quad (3.9b)$$

Usually a good estimate can be obtained by approximating the profile by two or three straight lines.

With (3.7b) one obtains

$$N \cong \frac{4\sqrt{2}}{3} \frac{f}{c_{min}^{3/2}} \left\{ \frac{h_1}{\Delta_1} \left| (\Delta_3+\Delta_b-\Delta_2)^{3/2} - (\Delta_3+\Delta_b-\Delta_1-\Delta_2)^{3/2} \right| \right. \quad (3.10)$$

$$\left. + \frac{h_2}{\Delta_2} \left| (\Delta_3+\Delta_b)^{3/2} - (\Delta_3+\Delta_b-\Delta_2)^{3/2} \right| + \frac{h_3}{\Delta_3} \left| \Delta_b^{3/2} - (\Delta_3+\Delta_b)^{3/2} \right| \right\} .$$

For $\Delta_b \gg \Delta_1, \Delta_2, \Delta_3$:

$$N \cong \frac{2\sqrt{2}\,f}{c_{min}^{3/2}} \left\{ h_1 (\Delta_3+\Delta_b-\Delta_2)^{1/2} + h_2 (\Delta_3+\Delta_b)^{1/2} + h_3 \Delta_b^{1/2} \right\} \quad (3.11)$$

$$\cong \frac{2\sqrt{2}\,f}{c_{min}^{3/2}} H \sqrt{\Delta_b} ,$$

where H is the total water depth. For the parameters chosen above one obtains 843 modes. For $\Delta_b = 0$ the result would be ($\Delta_1 \ll \Delta_3$):

$$N \cong \frac{4\sqrt{2}}{3} \frac{f}{c_{min}^{3/2}} \left\{ h_1 (\Delta_3 - \Delta_2)^{1/2} + \frac{h_2}{\Delta_2} \left[\Delta_3^{3/2} - (\Delta_3 - \Delta_2)^{3/2} \right] + h_3 \Delta_3^{1/2} \right\} \cong 215 \text{ modes.} \tag{3.12}$$

Two approximations of (3.12) are of special interest :

$$\Delta_3 = \Delta_2 : \quad N \cong \frac{4\sqrt{2}}{3} \frac{f}{c_{min}^{3/2}} (h_2 + h_3) \sqrt{\Delta_3}$$

and

$$\Delta_2 \ll \Delta_3 : \quad N \cong \frac{4\sqrt{2}}{3} \frac{f}{c_{min}^{3/2}} \left(\tfrac{3}{2} h_1 + \tfrac{3}{2} h_2 + h_3 \right) \sqrt{\Delta_3} .$$

Hence it turns out that the total number of modes lies approximately in the range

$$N = \left(1 \cdots \tfrac{3}{2}\right) \frac{4\sqrt{2}}{3} \frac{f}{c_{min}^{3/2}} H \sqrt{\Delta} \tag{3.13}$$

with Δ = maximum sound speed difference, H = total water depth.

To determine the average spacing between eigenvalues one only has to divide the difference

$$k_{max} - k_{min} = \frac{\omega}{c_{min}} - \frac{\omega}{c_{max}} \cong \frac{\omega}{c_{min}^2} (\Delta_2 + \Delta_b) \tag{3.14}$$

by the total number of eigenvalues as determined above. However, some care is necessary to obtain a useful result. It turns out that in the two regions

$$\omega^2/c_{max}^2 < k^2 < \omega^2/c_{min}^2 , \tag{3.15}$$

$$\omega^2/c_b^2 < k^2 < \omega^2/c_{max}^2 , \tag{3.16}$$

the eigenvalues are spaced very differently. For the first region to which the eigenvalues are confined when $c_b = c_{max}$ (i.e. $\Delta_b = 0$) the approximate structure of the eigenvalue spectrum can be found by calculating the number of modes as a function of the wave number for a two-layer profile (i.e. $k_1 = 0$). Then the total phase accumulated in the profile is given by the two Airy function arguments at $z = z_3$, i.e.

$$\Delta(z) = \Delta(z_3).$$

If one takes $\Delta(z_3)$ as a function of k varying between 0 and Δ_3 one can obtain the number of modes as a function of k as

$$n = \frac{4\sqrt{2}}{3} \frac{f}{c_{min}^{3/2}} \left\{ \frac{h_2}{\Delta_2} + \frac{h_3}{\Delta_3} \right\} [\Delta(k)]^{3/2}$$

with $\Delta(k)$ determined by

$$\frac{\omega^2}{c_{min}^2} - \frac{\omega^2}{[c_{min} + \Delta(k)]^2} = k_{max}^2 - (k_{max} - \delta k)^2$$

or $$\Delta(k) \cong (\tau_{min}^2/\omega)\,\delta k = (\tau_{min}^2/\omega)(k_{max} - k). \qquad (3.17)$$

Hence $$n \cong \frac{2\tau_{min}^{3/2}}{3\pi^{3/2} f^{1/2}} \left\{ \frac{h_2}{\Delta_2} + \frac{h_3}{\Delta_3} \right\} (\delta k)^{3/2} \qquad (3.18)$$

or

$$k_{max} - k(n) = \left(\frac{3}{2}\right)^{2/3} \frac{\pi f^{1/3}}{\tau_{min}} \left(\frac{h_2}{\Delta_2} + \frac{h_3}{\Delta_3} \right)^{-2/3} n^{2/3} = a n^{2/3},$$

i.e. the wave number changes with the two-thirds power of the mode number. Differentiating with respect to the mode number yields

$$dk/dn = -\tfrac{2}{3}\, a\, n^{-1/3} \cong \Delta k/\Delta n.$$

For $\Delta n = 1$ one obtains the spacing between eigenvalues as a function of the mode number:

$$\Delta k(n) = \left(\frac{2}{3} f\right)^{1/3} \frac{\pi}{\tau_{min}} \left(\frac{h_2}{\Delta_2} + \frac{h_3}{\Delta_3} \right)^{-2/3} n^{-1/3}. \qquad (3.19)$$

For the region (3.16) the eigenvalue distribution is quite different as one can see by inverting (3.11) and differentiating it with respect to n, after replacing Δ_6 by (3.17):

$$n(k) = \frac{2\sqrt{2}\, f}{\tau_{min}^{3/2}} H \left[(\tau_{min}^2/\omega)(k_{max} - k) \right]^{1/2}, \qquad (3.20)$$

$$k_{max} - k = (\pi \tau_{min}/4 f H^2)\, n^2,$$

with the result:

$$\Delta k(n) = -(\pi \tau_{min}/2 f H^2)\, n, \qquad (3.21)$$

i.e. in this region the spacing between eigenvalues increases linearly with mode number, whereas in the region $k > \omega/c_{max}$ it decreases slowly. Hence, the average eigenvalue spacing $\langle \Delta k \rangle$ is a useful quantity only in the latter region, since there the spacing remains relatively constant except for the first few modes. $\langle \Delta k \rangle$ can be defined for this case by dividing (3.14) with $\Delta_b = 0$ by (3.12):

$$\langle \Delta k \rangle = \frac{3\pi}{2\sqrt{2}} \left(\frac{\Delta_3}{c_{min}} \right)^{1/2} \left\{ h_1 \left(1 - \frac{\Delta_2}{\Delta_3} \right)^{1/2} + h_2 \frac{\Delta_3}{\Delta_2} \left[1 - \left(1 - \frac{\Delta_2}{\Delta_3} \right)^{3/2} \right] + h_3 \right\}^{-1} . \tag{3.22}$$

Taking as a special case $\Delta_2 = \Delta_3$ and $\Delta_2 \ll \Delta_3$, one gets

$$\langle \Delta k \rangle = \frac{3\pi}{2\sqrt{2}} \left(\frac{\Delta_3}{c_{min}} \right)^{1/2} (h_2 + h_3)^{-1} \qquad (\Delta_2 = \Delta_3)$$

$$\langle \Delta k \rangle = \frac{3\pi}{2\sqrt{2}} \left(\frac{\Delta_3}{c_{min}} \right)^{1/2} (h_1 + \tfrac{3}{2} h_2 + h_3)^{-1} \qquad (\Delta_2 \ll \Delta_3) .$$

Hence a good approximate guess is

$$\langle \Delta k \rangle = \frac{3\pi}{2\sqrt{2}} \left(\frac{\Delta_3}{c_{min}} \right)^{1/2} \frac{1}{H} \tag{3.23}$$

where H is the total water depth.

This expression is useful for establishing the step size for the wave number in the search for the eigenvalues. Upon some further manipulation one can obtain the expression

$$\frac{\Delta k(n)}{\langle \Delta k \rangle} \cong \frac{2}{3} \left(\frac{n}{N} \right)^{-1/3} . \tag{3.24}$$

From this one can see that near the highest mode (n = N) the actual spacing is 2/3 of the average, for n = N/8 i.e. for a very low mode the actual spacing is 4/3 of the average. Hence by using 6 x $\langle \Delta k \rangle$ as the step size one has approximately four steps between the highest modes, 16 steps between modes N/64 and N/64 + 1 which is usually sufficient to catch all modes except when nearly degenerate modes are present.

IV. SOLUTION OF THE CHARACTERISTIC EQUATION

As outlined in section 2, the eigenvalue problem (1.1) can be recast into the form (2.29) or (2.30) with the provision that the functions $u_p^w(z,\vec{\rho})$ and $u_p^b(z,\vec{\rho})$ or $u_{p\downarrow,\uparrow}(z,\vec{\rho})$ satisfy the wave equation (1.1a) and the boundary conditions (1.1b) and (1.1c). This requirement has been met by starting solutions of (1.1a) at the boundaries $z = 0$ and $z = \infty$, matching the boundary conditions there and propagating them layer by layer towards the joining point $z = z_B$ (for 2.29) or $z = z_M$ (for (2.30)).

The subscripts p on the solutions indicates that the particular functions describe the pth eigenmode corresponding to the eigenvalue k_p. Hence all the solutions u_p^w, u_p^b and $u_{p\uparrow}$, $u_{p\downarrow}$ can be considered implicit functions of a variable k which for the pth eigenfunction assumes the value $k = k_p$.

The task of solving the eigenvalue problem (1.1) is thus reduced to finding those values of k for which (2.29) or (2.30) are satisfied, more precisely to finding the zeros of the functions

$$W(k) = \left.\frac{\partial u^w(z,\vec{\rho},k)}{\partial z}\right|_{z=z_B} + \frac{\rho_w}{\rho_B}\gamma(\vec{\rho},k)\,u^w(z_B,\vec{\rho},k) \qquad (4.1)$$

or

$$W(k) = u_{\downarrow}(z_M, \vec{\rho}, k)\, u'_{\uparrow}(z_M, \vec{\rho}, k) - u'_{\downarrow}(z_M, \vec{\rho}, k)\, u_{\uparrow}(z_M, \vec{\rho}, k). \qquad (4.2)$$

As eq. (4.2) suggests, finding the eigenvalues corresponds to finding the zeros of the Wronskian between the down and up solutions at the joining point. Eq. (4.1) is essentially equivalent to (4.2) except that in this case use has been made of the fact that the up-solution is a simple exponential. This made it possible to factor out the up-solution and in fact drop it since it is nonzero everywhere.

An important property of the function W(k) is that it alternates signs between successive eigenvalues. For (4.1) this fact can be deduced in a straightforward way if this equation is interpreted as a matching condition of the derivatives of the down- and up-solutions:

$$W(k) = u^W(z_B, \vec{\rho}, k)' - u^B(z_B, \vec{\rho}, k)'$$

where subsequently the relation

$$u^B(z_B, \vec{\rho}, k)' = \gamma(\vec{\rho}, k)\, u^B(z_B, \vec{\rho}, k)$$

and the matching condition of the functions themselves

$$\rho_B\, u^B(z_B, \vec{\rho}, k) = \rho_W\, u^W(z_B, \vec{\rho}, k)$$

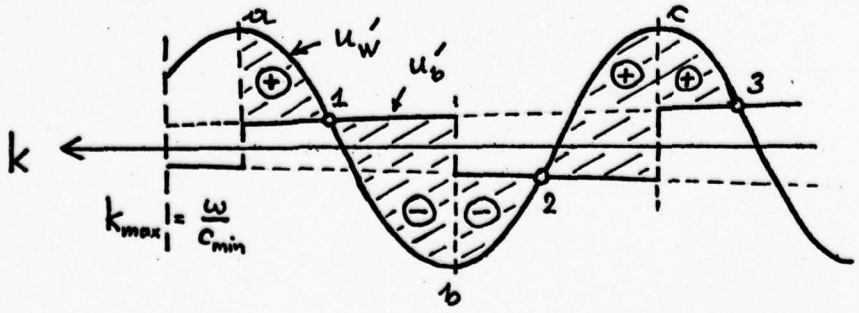
have been used.

It is well known that eigenfunctions of the type $u_p(z,\vec{\varsigma})$, regardless of their specific form, always have a standing wave type pattern of some sort with p nodes for the pth eigenmode. Hence if the functions $u_p^w(z,\vec{\varsigma})$ are always made to start out with the same slope at one boundary; e.g. the surface, they will have alternately positive and negative values at the other boundary. Therefore, since there are no modes within the ocean floor, $u^w(z_b, \vec{\varsigma}, k)$, and because of (2.23) also $u^b(z_b, \vec{\varsigma}, k)$, have different signs between successive eigenvalues. Hence the derivatives of $u^w(z_b, \vec{\varsigma}, k)$ as a function of k oscillates continuously between positive and negative values whereas the derivatives of $u^b(z_b, \vec{\varsigma}, k)$ is positive (negative) as long as $u^w(z_b, \vec{\varsigma}, k)$ is negative (positive) and it shows the discontinuous behavior indicated in Fig. 2, switching signs whenever $u^w(z_b, \vec{\varsigma}, k)$ changes sign (which is assumed to take place at points a, b, c etc. in Fig. 2). The plus and minus signs in Fig. 2 mark the areas where

$$u^w(z_B, \vec{\varsigma}, k)' \gtrless u^b(z_B, \vec{\varsigma}, k)'.$$

The numbered circles indicate the locations where W(k) changes sign, i.e. the locations of the eigenvalues.

Fig. 2



For eq. (4.2) an argument similar to the one just given applies if z_M was chosen such that the up solution is of the exponential type throughout. If both the down- and up-functions have oscillatory behavior at $z = z_M$ one can simulate their behavior there schematically by the functions

$$u_{\downarrow}(z,k) = A \sin \frac{k\pi}{z_B} z \qquad 0 < z < z_M$$

$$u_{\uparrow}(z,k) = B \sin \frac{k\pi}{z_B} (z_B - z) \qquad z_M < z < z_B$$

which have matching logarithmic derivatives at z_M whenever k is an integer. Then

$$\left[W(k)\right]_{z=z_M} = \frac{k\pi}{z_B} AB \sin k\pi ,$$

i.e., the Wronskian is zero for eigenvalues and alternates sign only there. This simple example can also be used to show that had one taken the condition of matching the logarithmic derivatives directly,

$$V(k) = \frac{u_{\downarrow}'}{u_{\downarrow}} - \frac{u_{\uparrow}'}{u_{\uparrow}} ,$$

instead of in the form of the Wronskian (4.2), the desirable property of the signs alternately only at the eigenvalues

would be lost:

$$V(k) \propto \frac{\sin k\pi}{\sin \frac{k\pi}{z_B} z_M \sin k\pi \left(1 - \frac{z_M}{z_B}\right)} .$$

While V(k) still has zeros only for k = integer the sign of V(k) now switches not only when an eigenvalue is passed (i.e. for k = integer) but also when one of the sines in the denominator has a zero.

Even though the arguments given above were not rigorous the qualitative conclusions apply directly to the general case of arbitrary eigenfunctions as long as these eigenfunctions are all non-degenerate. If this restriction is lifted and degenerate eigenmodes are possible the situation may become rather complicated and the simple method of finding the eigenvalues outlined in this section would certainly no longer work. However, it is very unlikely, as long as one is dealing with empirical sound velocity distributions, that one should encounter cases with truly degenerate eigenmodes. On the other hand, almost degenerate eigenmodes may occur occasionally in cases where the sound velocity distribution has pronounced local minima. The method of this section may then still be applied. However, a little care may be necessary to catch the sign change of W(k) between the almost degenerate eigenvalues.

For the purpose of this report it will be assumed that W(k) changes sign at every eigenvalue and only there. Then, in order to find the eigenvalues, i.e. those values of the k for which

$$W(k) = 0$$

is satisfied, one may simply use the following procedure. W(k) is calculated as a function of k with k changing in small enough steps and, whenever the result changes sign between two successive steps, one switches over to an iteration routine which locates the exact position of the zero. The range of discrete eigenvalues is restricted to

$$k_{max} > k > k_{min} . \qquad (4.3)$$

The lowest eigenmode is found for $k \approx k_{max}$. Hence the search is started at $k = k_{max}$ and proceeds by decreasing k steps of XKD; the quantity XKD has to be chosen properly. If it is too small, excessive amounts of computer time will be required to complete the search. If it is too large the search will locate only part of the eigenvalues. A good choice is usually

$$XKD \cong \langle \Delta k \rangle / 6 , \qquad (4.4)$$

where $\langle \Delta k \rangle$ is the average eigenvalue spacing as defined in section 3. If the bottom velocity c_B is much larger than c_{max}, the maximum sound velocity in the water, one may save consider-

able computation time by lettering XKD increase linearly with mode number for wave numbers

$$k < k_{min} = \omega / c_{max} .$$

Occasionally there are almost degenerate eigenmodes which are separated by only

$$\Delta k \cong \langle \Delta k \rangle / 100$$

or even less. In those cases it is usually still best to use (4.4) and catch the missed eigenvalues by a more detailed local search.

Formally eq. (4.1) and the method outlined for finding $u^W(z, \vec{\rho}, k)$ is sufficient to find all eigenvalues. In practice, however, several difficulties arise which have to be dealt with. Usually no problems arise for intermediate or high mode numbers since in these cases the function W(k) is sufficiently well-behaved near the sea floor $z = z_b$. The problems which do appear are basically caused by the fact that for low mode numbers $u^W(z, \vec{\rho}, k)$ has only a small oscillatory part near the sound speed minimum and two long exponential sections between there and the surface and the ocean floor, within which the function drops sometimes over several hundred orders of magnitude (see section 3). However, despite the exponentially falling appearance, both exponentially falling and exponentially rising components have to be present in the solutions to allow matching of the logarithmic derivatives at the boundaries. Very near the eigen-

values, the coefficients for the exponentially rising parts are of course extremely small so that the required exponentially falling appearance can be achieved. However for wavenumbers away from the eigenvalues matching at the boundary between the oscillating and the exponential part of u^W require large coefficients for the exponentially rising part. This way W(k) can assume very large values near the sea floor which may easily exceed the capacity of the computer. And more seriously, since a zero of W(k) has to achieved essentially by cancellation between u^W and its derivative the loss of accuracy due to very near cancellation of very large numbers does not permit to obtain a solution. Even the use of double precision arithmetic is not sufficient in many cases. Four steps were taken to permit the calculation of the eigenvalues despite these difficulties:

(1) Instead of always matching the down- and up-solutions at the sea floor (i.e. using eq. (4.1)) the matching is sometimes carried out at some appropriate boundary between two layers in the water (i.e. eq. (4.2) is used). This makes W(k) much more well-behaved for the low mode numbers.

(2) In the cases which up-layers are introduced the solutions increase very rapidly between the sea floor and the matching point. To prevent exceeding the capacity of the computer the function $u_\uparrow(z, \vec{\rho}, k)$ and its derivative $u'_\uparrow(z, \vec{\rho}, k)$ are scaled down at each layer boundary in steps of 10^{10} until $u_\uparrow < 10^{-5}$.

(3) If the solution is of the exponential type over the whole extent of a wide layer the propagation of the solution between the two boundaries becomes inaccurate if it is carried out in one step. In these cases the layers are divided into sublayers and the solutions are only propagated from one sublayer boundary to the next. This way the chances for errors due to near cancellation of large numbers is very much reduced.

(4) If the Airy function arguments are large and positive the corresponding values for the Airy functions can easily exceed the capacity of the computer. In these cases the Airy functions can be approximated by

$$Ai(\zeta) \propto f(\zeta) e^{-\frac{2}{3}\zeta^{3/2}}$$

$$Bi(\zeta) \propto g(\zeta) e^{\frac{2}{3}\zeta^{3/2}}$$

with corresponding expressions for the derivatives, where ζ may be of the order of 100 or even more.

Fortunately it turns out that one can find a formulation of the problem (see eqs. (2-15) and (2-16) where these very large or very small numbers are never needed individually but only in pairs in such a way that each pair contains one positive and one negative exponent. Hence, by carrying $f(\zeta)$, $g(\zeta)$ etc. and the exponents along separately one can avoid

overflows and underflows since the exponents appearing in the results are always only the small differences of two large numbers.

The computer code described in this report uses the method for solving the characteristic equation outlined earlier in this section. With the additional features just discussed it can be used to solve a wide range of practical problems. Propagation problems in the deep ocean ($H \sim 4000$ m) can be solved for frequencies up to several hundred Hz with high precision. For shallow water cases (water depth = a few hundred meters) the frequency range in which the program can be used extends up to several kHz. If a few precautionary measures are taken in preparing the profile data (see Section 7) then the range of application is basically only limited by the amount of computer time one is willing to spend.

The computer code consists of the main routine SEARCH and a set of subroutines. A detailed description of these programs is given in the next two sections. It should be noted that use has been made of this program in the following publication: A. Nagl, H. Überall, A. Haug and G. L. Zarur, "Adiabatic Mode Theory of Underwater Sound Propagation in a Range Dependent Environment", J. Acoust. Soc. Am. 63, 739 (1978).

V. DESCRIPTION OF THE COMPUTER CODE (MAIN PROGRAM)

Purpose of this section and the next is to discuss the set of computer codes written to solve the problem outlined in section 2 using the method described in section 4. The set consists of the main program SEARCH to which this section is devoted and the subroutines

PROP, LARGE, START, PR, LG, DAIRY, EXTREM, SINCOS, PARAM and SELAUT

which will be described in the next section.

The main program obtains input data from the subroutine PARAM, which provides mainly control parameters, and the file NFILE(1) in which the sound velocity profiles are stored. The inputs consist of the quantities (indices I refer to ith layer or ith layer boundary, indices J refer to jth profile):

NUMBER = number of boundaries for selected profile;
(X(I), Y(I), I = 1, NUMBER) = depths z_i and sound velocities $c(z_i, \vec{\rho})$ for profile selected;
CBOTM = sound velocity within the sea floor. These data come from the file NFILE(1), the following ones from the subroutine PARAM.

FREQ = sound frequency f

RHO1 = water density ρ_w

RHO2 = density of sea floor material ρ_B

XKD = step size of wave number k

IPRMIN = first profile to be investigated

IPRMAX = last profile to be investigated

XKST(J) = starting value of k for eigenvalue search for jth profile

XINCR1, XMAX1 = depth increment and maximum depth at which depth functions are to be calculated

NOPT = parameter to indicate whether or not the depth functions are to be calculated at a second set of depths. (NOPT = 1, NO; NOPT = 2, YES)

XINCR2, XMAX2 = depth increment and maximum depth for second set of depth function values (needed only if NOPT = 2).

NFILE(K) = Names of data files

NFILE(1) = Profile data file

NFILE(2) = Intermediate file

NFILE(3) = Depth function values at source depth

NFILE(4) = Depth function values at receiver depth

NFILE(5) = Eigenvalues

NFILE(6) = Wave functions and derivatives at layer boundaries

NFILE(7) = Intermediate file

JMIND = maximum number of up-layers allowed (once the down-solution has reached this layer a switch to the up-solution part may occur.)

ICT = maximum number of eigenvalues allowed for this run.

NPPL(I) = number of depth function values in ith layer.

ISLAY = number of points at which the depth functions are calculated (dimension of first depth function array).

ISOU = index of element in depth function array for source depth.

IREC1 = index element in depth function array for receiver depth for those modes for which there are no up-layers (if there are up-layers the index is called IREC and is computed in the program).

IRLAY = dimension of second depth function array for those modes for which there are up-layers (needed only if NOPT = 2).

IRLAY1 = dimension of second depth function array for modes without up-layers (needed only if NOPT = 2).

In executing the main program a number of essential steps can be distinguished which are described below. In order to facilitate associating variables used in the previous section with the corresponding symbols in the computer code both labels are sometimes given, connected by an arrow. In the program listing provided in the appendix the instruction, or set of instructions, each of the following steps is concerned with, are marked by arrows or brackets and are labeled by the appropriate step numbers.

(1) Control parameters are read in from PARAM

(2) A profile is selected and the data describing it are read in.

(3) The wave number k → XK for the first iteration is determined. Generally the starting value is ω/c_{min}. However, if for a particular profile a value XKST $< \omega/c_{min}$ is provided the iterations will start with this value.

(4) This step determines the slopes S(I) and the internal derivations for each layer. All quantities are multiplied with powers of H such that they come out dimensionless (H = total water depth).

(5) The search for the first eigenvalue is started at deterrent number 890. The program returns to this point each time the calculations for a particular eigenvalue have been completed in order to initiate the search for the next eigenvalue.

(6) At statement number 990 the wave number k → XK is reduced in steps of Δk → XKD. For each value of k thus selected the program proceeds to calculate the Wronskian W(k). This process continues until W(k) has different signs for two subsequent values of k. Then the value of k is determined by a different part of the program as described below. Control of the value of k remains there until k has converged to an eigenvalue.

(7) Calculation of W(k), whatever the value of k, is commenced at the statement number 994. First γ → UK is calculated. This is up to a minus sign the logarithmic derivative of

the function $u_p^B(z)$ inside the sea floor.

(8) Then in a DO-loop which extends to statement 3 the solution in the water is calculated by starting at the surface with the boundary conditions $u_p^w(0) = 0$ and $u_p^w(b)' = |\alpha|/\pi$ and propagating the function layer by layer down towards the sea floor. The solution and its derivative at the end of a layer (UOUT, UPOUT) in terms of the solution and its derivative at the beginning of the layer (UIN, UPIN) are calculated in one of the three subroutines START, LARGE, PROP. PROP is chosen if the argument $\zeta \to$ DX of the Airy functions is smaller or equal DM, which is chosen to be 11. LARGE is called if the argument is larger than DM. START is used for the first layer regardless of the argument. However, START can call LARGE if the argument is larger than 15 in at least part of the first layer. Similarly PROP will call LARGE if the argument is $\leq$ 11 at the beginning of the layer but grows $>$ 11 somewhere within the layer. Conversely, if the argument drops from $>$ 11 to $\leq$ 11 somewhere within a layer, LARGE calls PROP at this point to complete the layer.

This process is carried through all the way to the sea floor unless at the start of the calculations for some layer it is found that the argument of the Airy functions at the end of the layer is larger than 7, and that in addition the layer number J is larger than JD,

the number of completed down layers, which can be set as a parameter. In this case control of the program is transferred to another section of the program which is described in the next two paragraphs.
If this transfer does not occur and the propagation of the solution through all the layers is completed the program proceeds to calculate $u_p^B(z_B) \rightarrow$ SCALE and $W(k) \rightarrow$ FUNC which in effect corresponds to the Wronskian of the sea floor as outlined in section 4, and then skips to statement 45 in order to calculate the overall normalization constant (see paragraph 12.) The program actually assumes $u_p^B(z_B) = 1$ and $u_p^B(z_B)' = -\gamma$ with $\gamma \rightarrow$ UK. In this sense SCALE is the mismatch between $u_p^w(z_b)$ and $u_p^B(z_B)$. Since $u_p^w(z)'$ is calculated as the derivative with respect to the argument of the Airy function (see Section 2) and not with respect to z, $u_p^B(z_B)' \rightarrow$ UPBOTM has to be divided by the internal derivative at $z = z_B$ (= -AL) before it can be used to calculate the Wronskian. The additional factor is contained in the definition of UPBOTM to keep the derivative dimensionless. This is also necessary since $u_p^w(z)'$ is dimensionless by virtue of the fact that the argument of the Airy functions is defined to be dimensionless.

(9) The section of the program described in this and the next paragraph are entered if the arguments of the Airy fns.

grow too large in the lower layers of the profile for a reliable solution of the eigenvalue problem to be possible with the procedure outlined in the previous paragraph. If this part of the program is entered the result of the downward section of the calculation (UIN, UPIN) is kept for later use and a new solution is started at $z = \infty$. The coefficient of the solution within the sea floor is chosen such that the value of the solution at the sea floor is

$$u_p^w (z_B) = XFACT * 10^{-25}.$$

The quantity XFACT is at first set equal to one. Later it is adjusted to provide a match of the upward part of the solution with downward part. Since $u_p(z)$ for $z \geq z_B$ is an exponential the derivative at the sea floor is simply:

$$u_p^w (z_B)' = - UK * XFACT * 10^{-25}.$$

In the program this result is divided by (-AL) in order to give the derivative with respect to the argument of the Airy functions rather than with respect to z. The reason for this was explained in section 2. The factor H (=total water depth) which also appears is included to keep the derivative dimensionless.

Hence, because of conditions (2.23) and (2.24), the boundary values for the upward solution at the sea floor are

$$UDIN = RHO2/RHO1 * XFACT * 10^{-25}$$

$$UPDIN = (-UK) * H/(-AL) * XFACT * 10^{-25}.$$

Starting with these values the solution in the water, $u_w(z)$, is then propagated upward layer by layer in a DO-loop which extends to statement #4 whereby the solution and its derivative at the end of a layer, i.e. the upper boundary of the layer (UDOUT, UPDOUT) are calculated in terms of the solution and its derivative at the beginning, i.e. the lower boundary, of the layer (UDIN, UPDIN). To carry out this calculation the subroutines PR and LG are called if the arguments of the Airy function are $\leq$ 11 and $>$ 11 respectively. Transfers between PR and LG within a layer can occur analogous to the procedure outlined in paragraph 8.

(10) Since for the lower modes the upward solution may grow very rapidly over many orders of magnitude ($\propto \exp 2/3\,\xi^{3/2}$ sometimes $\xi \sim 50$ or even more) it was found necessary to provide the possibility of scaling down the solutions and their derivatives at the end of each upward layer. Generally, the downscaling consists of reducing u and u' in steps of 10^{10} until $|u| < 10^{-5}$. No scaling is performed in the uppermost up-layer. To keep track of the number of scalings at the end of each layer, this number is stored in the array ISC(J). This array is updated for each new value of k. However, once the Wronskian has changed signs the number of scalings at each boundary is kept constant in order to prevent the scaling pro-

cedure from disrupting the iteration process which finds the k for which

$$W(k) = 0.$$

If the number of scalings in two successive iterations were allowed to be different the convergence of the iteration process would be uncertain.

The fact that the quantity XFACT may change its value adds another complication. XFACT corresponds to the mismatch between the down-and up-solutions at their joining point. Until k has converged to some eigenvalue k_p, XFACTis actually kept equal to one. This does not interfere with finding the eigenvalue. However, in order to get a continuous solution and to find the correct value for the normalization integral it is necessary at the end to recalculate the solution with the eigenvalue just found and with XFACT assuming its proper value.

In this calculation the number of scalings in each layer may be different than before. Hence in order to keep track of the number of scalings in the final iteration another array has to be introduced: MSC(J). If the solutions are required at two different set of depths (NOPT = 2) an additional iteration is carried out where the number of scalings to be performed in each layer is taken from MSC(J).

(11) After the calculation for all the up-layers has been completed the Wronskian at the layer boundary where the up- and down-solutions join, is determined. If the program is in the phase where an eigenvalue has been found in the previous iteration (MFLAG = 1) two additional calculations are carried out in this section. First, the array NSC(ICOUNT, K) is determined which gives the total number of scalings which have been carried out on the solution with mode number ICOUNT between the sea floor and the Kth layer. In other words, in order to obtain the correct values for the solution for the mode p → ICOUNT in the Jth layer, the solution as stored in the file at this point has to be multiplied by

$$10^{10*NSC(ICOUNT,K)}.$$

The second operation carried out in this section concerns the contributions of the up-layer to the normalization integral. They are also affected by the scalings and have to be corrected. XS(J) and XT(J) are contributions to the normalization integral accumulated in layer J while the solution was calculated in subroutines PR and LG, respectively. It was assumed that only the uppermost up-layer (J=JX) contributes significantly to the normalization integral. The solution can be assumed to be exponentially decaying for at least a considerable part of this layer so that in the lower layers the square of the solution will be

insignificantly small. Hence, the contribution of all the up-layers to the normalization integral was taken to be

$$XNORM2 = (XS(JX)+XT(JX)) * 10^{-20*NS}$$

where NS is the difference

$$ISC(J) - MSC(J)$$

assumed over all up-layers.

(12) The contributions of the down-layers to the normalization integral are accumulated through the variable XNORM, each layer adding the quantity

$$\int_{z_n}^{z_{n+1}} |u_p^W|^2 \, dz.$$

The contributions of the up-layer are calculated similarly, added up to yield XNORM as outlined in the previous paragraph. The calculation of these integrals is described in the section on the subroutines. The contribution of the solution inside the sea floor,

$$\int_{z_B}^{\infty} |u_p^B|^2 \, dz,$$

is called XNOR1. It is obtained by observing that $u_p^B(z)$ is an exponential with the exponent $-\gamma \rightarrow -XK$. Hence the overall normalization constant is obtained as

$$SQN = \left(\frac{XNOR1}{SCALE^2} + XNORM + XNORM2 \right)^{-1/2}$$

where the fact was taken into consideration that SCALE is the mismatch between $\rho_w u_p^w(z_B)$ and $\rho_B u_p^B(z_B)$. SQN is

stored in the file NTYPE(2) with FORMAT (D).

(13) If MFLAG = 1 (= eigenvalue found in previous iteration) and NOPT = 2 (= second set of depth functions requested) the program will return to statement #994 (see paragraph 7) to calculate the depth functions in the range

$$0 \leq z \leq XMAX2$$

in steps of XINCR2. These values are also stored in file NTYPE(2). The format in this case is FORMAT (D 20.8).

(14) If MFLAG = 1 then all calculations concerning a particular eigenvalue are finished when statement #892 is reached and the program returns from this point to statement number 890 (see paragraph 5) to start the search for next eigenvalue.

(15) The program enters this section only once for each eigenvalue, namely if it is found that the wave numbers for two successive iterations differ by less than 10^{-15}. This condition is used as the criterion that the search for the eigenvalue has converged. The value $k \rightarrow XK$ obtained is stored in file NTYPE(2) with FORMAT(D). If there are uplayers the mismatch between the down- and the up-solutions (XFACT) is calculated here and the program returns to statement number 994 (see paragraph 7) with MFLAG = 1 for one (NOPT = 1) or two (NOPT = 2) additional iterations in which the following quantities are calculated and stored (as already partially indicated above):

(i) contributions from each layer to the normalization integral

(ii) the values of the solution $u_p^w(z)$ at one or two different sets of depths.

(iii) the values of the solutions and their derivatives at each layer boundary.

(16) This section of the program determines the wave number k for the next iteration. If the Wronskian W(k) has not changed sign since the last eigenvalue was found (IFLAG = 0) the program returns to statement #990 where k → XK is reduced by Δk → XKD. If the Wronskian has changed sign between this iteration and the previous one, JFLAG is set equal to 1 and the wave number for the next iteration is calculated by quadratic interpolation:

$$k_{n+1} = k_n - \left[\frac{W(k_n)}{W(k_n) - W(k_{n-1})}\right]^{1/2} (k_n - k_{n-1}).$$

If JFLAG = 1 already, k for the next iteration is calculated by linear interpolation:

$$k_{n+1} = k_n - \frac{W(k_n)}{W(k_n) - W(k_{n-1})} (k_n - k_{n-1}),$$

and the program also returns to statement number 994.

(17) This section of the program is entered after all eigenvalues for a particular profile have been found. It essentially reads back all the data written into file

NFILE(1), pickes out the depth function values at the source and receiver depths and normalizes them. Then it stores the values at the source depth for all modes into file FILE (3), the corresponding values at the receiver depths into file NFILE (4), the eigenvalues into fileNFILE (5), the wave functions and their derivatives at the larger boundaries into file NFILE (6). In all cases FORMAT (86D) is used.

VI. DESCRIPTION OF THE COMPUTER CODE (SUBROUTINES).

Subroutine SELAUT

This routine transfers the data for the selected profile from the profile data file NFILE (1) to the temporary file NF1LE (2) converting them in the process into a format which is convenient for reading them to the main program SEARCH. This arrangement is convenient since the main program does not have to be changed if for some reason the arrangement of the data in the profile data field should be altered.

Subroutine PARAM

Purpose of this program is to provide a convenient input channel for all the input parameters except the profile data. Through the set of input parameters provided it is usually possible to set up the program for specific cases without having to change and recompile the main program and any of the other subroutines every time. The set of parameters provided is listed in the previous section, where the main program is described. Information on how to set up a run is given in section 7.

Subroutines DAIRY

It calculates the Airy functions and their derivatives to 14 significant digits for arguments in the range

$$-1000 < DX < 15 .$$

It is called from START, PROP and PR. The arguments for which the Airy functions are needed in these subroutines are usually smaller than DM, which is 11 in PROP and PR and 15 in START.

If the argument is larger than that value in some part of a layer the program switches over to LARGE or LG. The only time when the argument can be larger than 11 in PROP and PR is in the case of a layer in which the argument decreases from above 11 to below 11. Then the program starts the layer in LARGER or LG and then switches to PROP or PR at the beginning of the sublayer in which the argument decreases below 11. At this point the argument may be larger than 11 by the amount DA by which the argument may change within a sub-layer. Hence if one makes sure that $DA < 4$ the arguments for which DAIRY is used always stay below the upper limit for which DAIRY is defined.

For layers with very small slopes in the velocity profile the arguments may fall considerably below -1000 (see section 3). For these cases the

Subroutine SINCOS

is called which uses the asymptotic expansion for large negative arguments given in (3.1).

Subroutine EXTREM

Calculates the Airy functions and their derivatives for large positive arguments. This routine is called from LARGE and LG. The arguments for which the Airy functions are needed there are usually larger than 11. An exception are the calculations in a sublayer in which the arguments rise above or fall below 11, i.e. the sublayers at the beginning (the end) of which the switch from (to) the routines PROP, START or LG occurs. Then the arguments can be below 11 by the amount by which they can change within a sublayer.

This means that the ranges of arguments for which EXTREM is called extends from about 7 to possibly several hundred. To avoid overflows resulting from the corresponding large values for the Airy functions the results are given in 2 parts, one being the exponent of the asymptotic form (= ZETA), the other one the value of the polynomial

multiplying the exponential (see (3.1)). In order to take advantage of the high precision of the subroutine DAIRY for the smaller arguments, this routine is called by EXTREM to calculate the Airy functions for arguments ≤ 15.

Purpose of the remaining five subroutines (PROP, PR, START, LARGE, LG) is to carry out the following functions: (1) To provide the depth functions and derivatives at the end of an interval in terms of the depth function and its derivative at the beginning of the interval. START, PROP and LARGE are used for down-layers, and PR and LG for up-layers. (2) If MFLAG = 1, to calculate the integral of the square of the depth function over the range of the layer. (3) If MFLAG = 1, to calculate the depth function at integer multiples of XINCR. The inputs to all 5 of these routines are with a few exceptions:

AL, BE quantities in terms of which the arguments of the Airy functions are defined. According to (2.10) the arguments can be written as

$\zeta_{pi}(z,\vec{\rho}) = -\alpha_i(\vec{\rho})[z+\beta_i(\vec{\rho})]$ or $-AL*(BE+X)$,

where AL corresponds to the internal derivative $d\zeta/dz$ and BE to $d_{pi}(\vec{\rho})/s_i(\vec{\rho})$, with $d_{pi}(\vec{\rho})$ being defined by (2.9). AL and BE can always be assumed

to be constant within the range (X, XJ) of the interval. Hence the arguments of the Airy functions are always either a monotonically increasing or a monotonically decreasing function of the depth.

X, XJ — Depth variables defining the beginning and the end of the depth interval for which the calculations are performed.

U, UP — Depth function and its derivative at beginning of the interval.

XNEW — Depth variable which is changed in increments of XINCR inside the subroutines. It specifies the depth at which depth functions may be calculated and it is used to define the boundaries of sublayers. The input value of this variable is always $\leq$ X for START, PROP, LARGE and $\geq$ X for PR, LG.

XINCR — Step size with which the depth variable XNEW may be changed.

TT — Ratio of internal derivatives at boundary between two intervals as defined in (2.13') (internal derivative for previous interval divided by internal derivative for this interval).

MFLAG Flag that is set in the main program SEARCH and transmitted to the subroutines through a COMMON statement. It indicates that an eigenvalue has just been found. In the subroutines it sets up the conditions for calculating the contribution to the normalization integral from the layer under consideration and, if in addition MFLAG1 = 1, for calculating the depth functions at prescribed depths.

MFLAG1 This flag is also transmitted to the subroutines through a COMMON statement. It is set to 1 in the main program SEARCH if the maximum depths (XMAX1 or XMAX2), down to which the values of the depth functions are requested, are within or below the layer under consideration. If the flag has the value 1 it causes the subroutines to calculate the depth functions for all values of XNEW which fall inside the interval, provided MFLAG also has the value 1.

The output from the subroutines consists of:

UJ, UPJ Depth function and its derivative at end of interval.

During execution of the subroutines this variable has been increased (START, PROP, LARGE) or decreased (PR, LG) by an integer multiple of XINCR from its input value. Its output value is always $\leq$ XJ for START, PROP, LARGE, and $\geq$ XJ for PR and LG.

XNORM This is the variable through which the contribution to the normalization integral of all the down-layers is accumulated, i.e. whenever any of the subroutines START, PROP, or LARGE is called and the condition MFLAG = 1 exists the integral of the square of the depth function within the layer boundaries is calculated and the result added to XNORM. The variable is transmitted to and from the main program via a COMMON statement.

X1SQ, X2SQ These are the contributions to the normalization integral calculated in the subroutines PR (=X1SQ) and PG (=X2SQ). PR and LG have both quantities as arguments since a transfer between the 2 subroutines can occur within a layer. The contribution of all the up-layers are added up in the main program.

Subroutines PROP and PR have a similar structure and are therefore discussed together. They consist essentially of 3 sections:

Section (1): Calculations at the beginning of the layer. The coefficients AJ, BJ of the Airy functions defining the depth function in this layer are determined here following eqs. (2-13'). If MFLAG = 1 the value of the normalization integral at the start of the integration interval (=BEG) is also calculated at this point.

Section (2): Calculations at sublayer boundaries. This section is entered in two cases:

(a) If the argument of the Airy function at XJ, the end of the layer, is > 11.

(b) If MFLAGi = 1, i.e. if the calculation of the depth function at points within this layer is requested. In both cases sublayer boundaries are established within the layer at regular intervals, starting with the input value of XNEW increased for PROP (decreased for PR) by XINCR and then proceeding in steps of XINCR. At each boundary the values of depth function (WF) and its derivative (WP) are calculated. In case (2) the depth function at each sublayer boundary is stored in file NFILE (2) with FORMAT (D20.8). In case (1) the program proceeds from one sublayer to the next until at some sublayer boundary the argument of the Airy function exceeds

11. Then subroutine LARGE is called using as inputs

X = XNEW

U = WF

UP = WP

TT = 1 (since the internal derivatives do not change across the sublayer boundaries),

completing there the remainder of the calculation for this layer. After completion of the CALL statement the program immediately returns to the main program. If MFLAG = 1 the value of the normalization integral at the latest XNEW is calculated (-CEN) and the quantity (-CEN -BEG) is added to XNORM before returning to the main program, i.e. the contribution to the normalization integral of the region between the beginning of the layer and the XNEW at which the transfer to LARGE occurs as added to the total.

Section (3): Unless a transfer to subroutines LARGE occurred in Section 2 the calculations in PROP and PR are wound up by calculating the depth function and its derivative at the end of the layer (i.e. UJ, UPJ) and if MFLAG = 1 by calculating the normalization integral at the end of the layer (-CEN) and adding the difference to XNORM. If Section 2 was skipped (i.e. if the argument of the Airy functions was smaller than 11 for the whole layer and if MFLAG1 was 0, so that the depth functions did not

have to be calculated inside this layer), then XNEW still has its initial value and it has to be updated. If the argument of the Airy functions is $\leq$ 11 at the beginning of the layer but $>$ 11 at the end, only the solution for the first part of the layer is calculated in PROP (or PR) up to the last sublayer boundary at which the argument is less than 11. The solution for the rest of the layer is then calculated in LARGE (or LG) as indicated above. The argument at which the transfer occurs may be from very slightly less than 11 to considerably less than 11. The latter is the case if the slope is large and the argument at the next sublayer boundary is just above 11. The reverse case where the argument is $>$ 11 at the beginning of the layer and $\leq$ 11 at the end, i.e. when the calculations for the first part of the layer are made in LARGE (or LG) and then the transfer is made to PROP (or PR) is discussed in the section on LARGE and LG.

Subroutine START is called to perform the calculations in the layer nearest the surface. It contains 4 sections: the first three are used if the argument of the Airy functions remain below 15 for the whole layer, the fourth one if the argument exceeds 15 for at least part of the layer.

Section 1: calculations at the surface. Calculation of the coefficients to the Airy functions according to assumptions (2-18) and (2-19). If MFLAG = 1 the value of the normalization integral at the surface is also determined here.

Section 2: calculations at the end of the layer. This section determines the depth function and its derivative at the end of the layer and, if MFLAG = 1, the contribution ZNORM of the layer to the normalization integral XNORM.

Section 3: updates XNEW to the largest value $\leq$ XJ and for MFLAG1 = 1 calculates the depth function inside the first layer at intervals XINCR and writes them into file NFILE (2).

Section 4: This section is used to transfer the calculations for the entire first layer to LARGE. This is done if anywhere inside the layer the argument of the Airy function exceeds 15. For the part of the layer for which the argument is $\leq$ 15 the calculation should actually be done in START but it is assumed that if the argument is $>$ 15 somewhere in the layer it is not much less anywhere else. This is a reasonable assumption since surface layers with large arguments are never very wide. So the argument may be large but it cannot vary much over the extent of the layer.

Subroutines LARGE and LG again have similar structure and may therefore be discussed together. These subroutines are used for those intervals in which the Airy function arguments are large ($>$ DM). Such an interval may extend over a whole layer or over

part of a layer. Since the Airy function arguments are monotonically increasing or decreasing within a layer, it is clear that of the arguments at the beginning (= DX) and at the end (= DX1) at least one must be larger than DM, if the routines LARGE and LG are involved in the calculation for a particular layer, and one can distinguish 3 cases:

(1) DX > DM and DX1 > DM. The layer is calculated entirely by LARGE or LG (interval extends between the layer boundaries X and XJ.)

(2) DX > DM but DX1 ≤ DM. First part of layer is calculated in LARGE or LG (interval extends from X to value of XNEW at switch. For calculations of remainder of layer PROP or PR are called.

(3) DX ≤ DM, DX1 > DM. First part of layer is calculated in PROP or PR. For calculations of remainder of layer LARGE or LG are called by PROP or PR (interval extends from value of XNEW at switch to XJ).

All intervals for which LARGE and LG are used are divided up into sublayers of width XINCR, and the depth functions are propagated from one sublayer boundary to the next using eqs. (2-15). The first sublayer boundary extends from X (= interval boundary, which is the layer boundary in cases (1) and (2) and the value of XNEW at the switch in case (3)) to $XNEW_{input}$ + XINCR, the first

sublayer boundary within the interval. Subsequent sublayers are established by changing the depth variable XNEW in steps of XINCR and using these values as boundaries. The last sublayer extends from the highest (lowest) value for XNEW which is $\leq$ XJ ($\geq$ XJ), or in case (3) to the first value of XNEW for which the Airy function argument is <11.

Both LARGE and LG essentially consist of 3 sections:

Section (1): At the beginning of the interval only the Airy functions for the argument at this point are calculated.

Section (2): Calculations for each sublayer. The solutions are propagated from one sublayer to the next by expressing the depth functions and their derivatives at the end of each sublayer (U1, U1P) in terms of the corresponding values at the end of the previous sublayer (U, UP) and the coefficients C_{ij} following eqs. (2-16).

If MFLAG = 1 the depth function value is written into the file NFILE (2) at every value of XNEW inside the interval.

If MFLAG = 1 the contribution of the sublayer to the normalization integral is calculated.

If at some sublayer boundary it is found that within the following sublayer the Airy function argument goes below DM, the calculations

are transferred to PROP or PR, in the case MFLAG = 1 after the contribution of all the sublayers up to this point to the normalization integral is calculated and added to XNORM.

The argument at which the transfer occurs may vary from slightly larger than 11 to considerably larger than 11. The latter case may happen if the argument changes very much over the extent of the sublayer and it is only slightly less than 11 at the next sublayer boundary. This means that the Airy function argument could be larger than 11. This means that the Airy function argument could be larger than 15 for calculations in PROP or PR. This could cause problems since the subroutine DAIRY which is used in PROP and PR for calculating the Airy functions is accurate only for arguments below 15, and more seriously, the value of the Airy functions may exceed the capacity of the computer if the argument is allowed to go too high. In order to avoid these problems the width of the sublayers should be chosen small enough to keep the change in the Airy function arguments within a sublayer below 4.

According to (3.6) the difference in the Airy function arguments can be written as

$$\Delta\zeta = -(2^{1/3}/\tau_{min})\left(\frac{\omega}{\Delta_i/h_i}\right)^{2/3}\left[\Delta(z_{n+1}) - \Delta(z_n)\right]$$

where

$$\Delta(z_{n+1}) - \Delta(z_n) = (\Delta_i/h_i)(z_{n+1} - z_n) = (\Delta_i/h_i)\times INCR$$

i.e. the difference in the Δ's is just the sound velocity slope in the layer times the thickness of the sublayer. Hence

$$\Delta\zeta_{SL} = -\frac{2^{1/3}}{r_{min}}\,\omega^{2/3}\left(\frac{\Delta_i}{h_i}\right)^{2/3} XINCR .$$

So in order to keep the Airy function arguments from exceeding 15 for calculations in PROP and PR, XINCR should be chosen such that

$$\frac{2^{1/3}}{r_{min}}\,\omega^{2/3}\left|\frac{\Delta_i}{h_i}\right|^{2/3} XINCR < 4 \qquad (6.1)$$

for all layers.

Section (3): Calculations at the end of a layer consist of propagating the solution from the last sublayer boundary to the end of the layer and if MFLAG = 1 of calculating the contribution of the whole interval to the normalization integral and adding it to XNORM.

VII. INSTRUCTIONS FOR USING THE COMPUTER CODE

This section outlines the steps necessary to obtain the solutions to the eigenvalue problem (1.1) using the computer code described in the previous two sections.

(1) Preparation of input parameters (entered into PARAM).

(NFILE(J), J = 1.5)	Naming of files: J = 1 profile file J = 2 temporary file J = 3 depth functions at source depth J = 4 depth functions at receiver depth J = 5 eigenvalue file J = 6 file for depth functions and derivatives at layer boundaries
FREQ, RH01, RH02	Sound frequency, water density, sea floor density
IPRMIN, IPRMAX	First and last profile to be investigated
XKST(J)	Starting value of k for eigenvalue search for jth profile. Ordinarily the values of this array are all set equal to 1000. This way the first mode to be found will be the one with mode number one. If desired a run can be started with a higher mode number by specifying the appropriate

values of the wave numbers XKST for each of the selected profiles.

ICT — This parameter specifies the number of modes to be calculated. If a number >N (N = maximum number of modes for a profile) is selected all modes are calculated.

JMIND — Maximum number of up-layers allowed. Usually a good choice is to take one less than the number of layers below the location of the sound speed minimum.

XKD — Step size of wave number during search for sign change of W(k). This quantity determines to a large extent the amount of computer time required for a particular profile. A good value to start with is XKD = $\langle \Delta k \rangle /6$ with $\langle \Delta k \rangle$ as defined by (3-23). For the region

$$k < \frac{\omega}{c_{max}} \qquad c_B - c_{max} \gg c_{max} - c_{min}$$

the spacing between eigenvalues increases linearly with mode number according to (3.21) Hence, if

$$c_B - c_{max} \gg c_{max} - c_{min}$$

with, e.g.,

$c_{min} \cong 1490$ m/s, $c_{max} \cong 1540$ m/s,

$c_B \cong 1800$ m/s,

the spacing between eigenvalues can grow very large and one can save a considerable amount of computer time if the step size XKD is made to increase linearly with mode number when searching for eigenvalues in the region $k < \frac{\omega}{c_{max}}$ This can be done by making use of (3-21), increasing XKD by

$$\frac{\pi c_{min}}{2fH^2}$$

each time an eigenvalue has been found.

XINCR1, XMAX1
XINCR2, XMAX2

These quantities specify the depth at which the depth functions are calculated. In addition XINCR1 and XINCR2 determine the widths of the sublayers.

NOPT

If the source and receiver depths are related by an integer or if they have a convenient common factor the depth functions at both locations can be calculated with one pass through the program after a particular eigenvalue has been found. In this case one sets

NOPT = 1

and XINCR2, XMAX2 are not needed.
XINCR1 is chosen such that both the source and receiver depth are an integer multiple of it and XMAX1 is taken to be the source or receiver depth, whichever is larger (e.g., source depth = 200m, receiver depth = 500m, XINCR1 = 100m, XMAX1 = 500m).
If the source and receiver depth have no convenient common factor (e.g. sorce depth = 29m, receiver depth = 100m) the depth functions at these depths are calculated at two different passes through the program.
In this case

NOPT = 2

XINCR1= XMAX1 = Source depth

XINCR2 = XMAX2 = Receiver depth.

The option NOPT = 1 must also be used if the receiver position falls into an up-layer. This case occurs when the water depth is very large and the receiver is located near the ocean floor. At the end of section 6 it was mentioned that there is a constraint on XINCR, which took the form (6.1). Accordingly XINCR1 and XINCR2 have to satisfy the condition

$$XINCR1,\ XINCR2 < 1400\, f^{-2/3} (\Delta/h)_{max}^{-1/3}$$

where f is the frequency and $(\Delta/h)_{max}$ the maximum sound speed gradient of profile.

NPPL(I), ISLAY, ISOU, IREC1, IRLAY, IRLAY1 are the parameters needed to extract the desired output data (eigenvalues, normalized depth function at source and receiver and the depth function and their derivatives at each layer boundary) from the temporary output file NFILE(2).

If NOPT = 1 only the first 4 parameters are needed. NPPL(I) gives the number of data points in each layer (the surface is not counted as a data point, data points at layer boundaries belong to the preceeding layer), i.e. if XINCR1 = 50 and the widths of the first and second layer are 100m and 240m, NPPL(1) = 2 and NPPL(2) = 5, etc.

Data are taken at intervals of XINCR1 to the largest multiple of XINCR1 which falls into the layer in which XMAX1 lies. ISLAY is the total number of data points taken, ISOU and IREC1 specify the data points which correspond to the source and receiver depths respectively. If e.g. the source depth is 100m, the receiver depth 150m, in the example given above, then

ISLAY = 7 (= NPPL(1) + NPPL(2))

ISOU = 2

IREC1 = 3

If NOPT = 2 the depth functions at the receiver depth are obtained in a second

loop through the programs as explained above. If the receiver position does not fall in an up-layer for any of the modes one sets

IRLAY = IRLAX1 = total number of data points taken.

If the receiver depth does fall into an up-layer for some modes then it is necessarily very large and one sets

XMAX2 = total water depth H.

Then

IRLAY1 = total number of data points taken ($\lesssim$ H/XINCR2)

and

IRLAY = IRLAY1 if the ocean bottom is not a data point,

IRLAY = IRLAY1-1 if the ocean bottom is a data point.

(2) Preparation of profile data file. The data essentially consist of the array (1.2) plus the corresponding depth values Z_i. Sometimes the sound velocities are not given as the same depth for all profiles. Therefore the depth values are also taken as a function of range.

The profile data file is read by the subroutine SELAUT. At present the following data structure is assumed:

1st line: N = Number of layer boundaries for first profile, FORMAT (I3).

2nd-jth lines: N + 1 velocity values, 5 to a line, FORMAT (5D14.7) = sound velocity at each layer boundary + bottom velocity for first profile.

(j + 1)st-kth line: N + 1 depth values, 5 to a line, FORMAT (5D14.7) = depth values of layer boundaries + bottom depth for first profile.

1st-kth line repeated for each additional profile.

If this arrangement is found inconvenient it can easily be changed. The only program affected by this change would be the short subroutine SELAUT.

The program in its general form does not accept isovelocity layers. From (3.6) one can see that for those cases the Airy function arguments become ∞ and (3.1) shows that the Airy functions are not defined there. Isovelocity layers do rarely occur in empirical profiles and if they do they can easily be avoided by replacing the vanishing slopes by very small slopes (e.g. .01 m /sec per 100m). If desired the program can

easily be generalized to include the (essentially trivial case) of isovelocity layers by adding 2 subroutines which propagate the down- and up- solutions for these cases between layer boundaries and which are called by the main program in place of PROP, LARGE or START and PR or LG.

Before setting up the profile data file it is necessary to check the widths of the vertical layers and to make sure they do not exceed a certain maximum value which is given by the requirement that the Airy functions should not grow or decay within one layer by more than the maximum number the computer can handle. The program in its present form was written for the PDP10 in which the exponents are limited to $\lesssim \pm 35$. At the end of each up-layer the depth functions are scaled down to $\lesssim 10^{-5}$. Hence the maximum allowable growth within one up-layer is 10^{40} (or 10^{60} for the bottom layer where the depth function is made to start out with $\sim 10^{-25}$). Layers which do not satisfy this requirement have to be subdivided. An estimate for the maximum allowable layer width can be obtained by using the formulas derived in section 3. From there it follows that the Airy function arguments within the lowest layer (which is usually the one causing trouble) is

$$\zeta(z) = (2\omega^2 \Delta_i/h_i)^{1/3} (z_B - z)/c_{min}.$$

The ratio of the Airy function values at 2 locations within that layer is

$$\frac{Ai(\zeta_2)}{Ai(\zeta_1)} \sim e^{\frac{2}{3}\left[\zeta(z_1)^{3/2} - \zeta(z_2)^{3/2}\right]}$$

$$= e^{\frac{4\pi}{3}\sqrt{2} f (\Delta_n/h_n)^{1/2} (h_n/c_{min})^{3/2}\left[\left(\frac{z_B - z_1}{h_n}\right)^{3/2} - \left(\frac{z_B - z_2}{h_n}\right)^{3/2}\right]},$$

where Δ_n/h_n is the sound speed gradient and h the width of the layer. For the example chosen in section 3 (f = 250 Hz, Δ_n/h_n = .015 1/3 and h_n = 3000 m):

$$\frac{Ai(\zeta_2)}{Ai(\zeta_1)} \sim 10^{223\left[\left(\frac{z_B - z_1}{h_n}\right)^{3/2} - \left(\frac{z_B - z_2}{h_n}\right)^{3/2}\right]}.$$

With $Z_2 = z_B$ = 4000 m, Z_1 = 1000 m as in the example one obtains

$$\frac{Ai(\zeta_2)}{Ai(\zeta_1)} \sim 10^{223}$$

as before (see section 3). Hence in this case the 3000 m wide bottom layer would have to be divided up into about 6 separate layers of ~ 500 m each.

The example presented is actually something like an extreme case since 250 Hz for a deep ocean case is about the upper limit

of application for the normal mode theory and furthermore the layer thicknesses as they are usually given rarely exceed 1000 m. Hence only in rare cases do layers have to be divided up. The program presented in this report was written for the PDP10 in which the exponents are limited to $\sim \pm 35$. For a machine with a larger exponent range the restrictions on the layer widths can of course be relaxed correspondingly.

(3) Adjustment of array dimensions.

The maximum number of layers in any of the profiles determines the dimension of the arrays

ISC, MSC, S, B, X, Y, T, TD, C, P, ALPH, XS, XT, NPPL,

and the second dimension of

NSC.

The dimensions of these quantities should be chosen as the number of layers + 2.

The dimension of the arrays

JUPA, IRECA,

the first dimension of

NSC

and the second dimensions of

UO, UF, XKA

are determined by the maximum number of modes obtained for any particular profile.

The array

XKST

has to be dimensioned according to the number of profiles. Dimensioned arrays are limited to the main program, except for

XKST and NFILE

which appear both in the main program and in the subroutines PARAM and

X and Y

which are used in SELAUT.

(4) Execution of program.

After the preparations described in (1)-(3) are taken care of the program can be executed. The following routines and files are involved:

Main program: SEARCH

Subroutines: PARAM, START, PROP, LARGE, PR, LG, DAIRY, EXTREM, SELAUT

Profile data file: NFILE (1)

The results of the calculations is output through the files

NFILE (3): depth functions at source depth

NFILE (4): depth functions at receiver depth

NFILE (5): eigenvalues

NFILE (6): depth functions and derivatives at layer boundaries

(5) Mode number test.

If the eigenvalue spectrum contains almost degenerate modes it is possible that the search procedure contained in the program skipped over some modes. Therefore after execution of the program one has to carry out a check to see whether the eigenvalue spectrum for each of the profiles is complete. This consists of rerunning the program for the highest mode for each profile (using the appropriate starting values XKST for the eigenvalue search) with a sufficiently small XINCR1 to obtain a complete picture of the modes and plotting the modes as a function of depth. The depth function values can be taken from the temporary file NFILE(2). If one finds that for a particular profile the number of modes corresponds to the mode number then one can conclude that the solution of the eigenvalue problem was successfully completed for this profile. If the number of modes is larger than the mode mumber this means that some modes are missing. The missing modes can be found by repeating the procedure described above for some lower modes (e.g. starting with a mode number $\gtrsim$ 1/4 of the highest mode number), until the mode number of the missing modes has been established. The exact location of the missing modes can then be found by running the program between the neighboring two modes with a sufficiently small XKD.

```
00000      C
00000      C                    SEARCH
00000      C
00000      C                  (MAIN PROGRAM)
00000      C
00010            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
00020            DIMENSION FU(100),EIG(100),ISC(60),MSC(60),JUPA(80)
00030            DIMENSION S(60),B(60),X(60),Y(60),T(60),TD(60),C(60)
00040            DIMENSION P(60),ALPH(60),UO(6,80),UF(6,80),XKA(6,80)
00050            DIMENSION XS(60),XT(60),NPPL(60),NSC(80,25),IRECA(80)
00060            DIMENSION XKSI(10),NFILE(7)
00070            COMMON /IFLAGS/ MFLAG,MFLAG1,MTYPE /JC/JCOUNT
00080            COMMON /SQNORM/ XNORM
00090            COMMON /DATCOM/XKST, FREQ,RHO1,RHO2,XKD,XMAX1,XMAX2,
00100        1    XINCR1,XINCR2,NPPL,JMIND,NFILE,ISLAY,IRLAY,
00110        2    ISOU,IREC,NOPT,IPRMIN,IPRMAX,ICT,IRLAY1,IREC1
00120            CALL PARAM                                          (1)
00130            OM=11.0D0
00140            XKOS=XKD
00150            MTYPE=NFILE(2)
00160            NTYPE=NFILE(2)
00170            DO 15 IPR=IPRMIN,IPRMAX
00180            CLOSE (UNIT=NFILE(1))
00190            CLOSE (UNIT=NFILE(2))
00200            CALL SELAUT(IPR,NFILE(1),NFILE(2))
00210            CLOSE (UNIT=NFILE(2))
00220            XNOR1=0.0D0
00230            SCALE=1.0D0
00240            SCALE2=1.0D0                                       (2)
14113 00250      DSCALE=1.0D20
00260            ICOUNT=0
00270            READ(NFILE(2),26) NUMBER
00280       26   FORMAT(I3)
00290            READ(NFILE(2),1) (X(I),Y(I),I=1,NUMBER)
00300            READ(NFILE(2),1) CBOTM
00310            CLOSE(UNIT=NFILE(2))
00320            CMIN=Y(1)
00330            DO 199 I=1,NUMBER
00340            IF(Y(I).LT.CMIN) CMIN=Y(I)
00350      199   CONTINUE
00360            NMIN=NUMBER-1
00370            H=X(NUMBER)
00380            JD=NMIN-JMIND
00390            H2=H*H
00400            PI=3.14159265358979323840D0
00410            EDIF=1.0D0
00420       1    FORMAT(2D)
00430            OM=2.0D0*FREQ*PI
00440            OM2=OM**2                                          (3)
00450            XK=OM/CMIN
00460            IF(XK.GT.XKST(IPR)) XK=XKST(IPR)
12 00470         P(1)=H2*OM2/(Y(1)**2)
11 00480         DO 43 I=1,NMIN
10 00490         X(I+1)=X(I+1)/H
 9 00500      43  P(I+1)=H2*OM2/(Y(I+1)**2)                          (4)
 8 00510         DO 44 I=1,NMIN
 7 00520         S(I)=(P(I+1)-P(I))/(X(I+1)-X(I))
 6 00530         C(I)=P(I)-S(I)*X(I)
 5 00540      44 CONTINUE
 4 00550         ITER=0
 3
```

```
00560            TH=1.0D0/3.0D0
00570            DO 66 I=1,NMIN
00580            SN=S(I)/DABS(S(I))
00590    66      ALPH(I)=SN*(S(I)**TH)
00600            T(1)=S(1)/DABS(S(1))
00610            DO 67 I=2,NMIN
00620            T(I)=ALPH(I-1)/ALPH(I)
00630    67      CONTINUE
00640    890     IFLAG=0
00650            ICAL=0
00660            XMAX=XMAX1/H
00670            XINCR=XINCR1/H
00680            IF(ICOUNT.EQ.ICT) GO TO 980
00690            MFLAG=0
00700            MFLAG1=0
00710            XNORM=0.0D0
00720            JTER=0
00730            KFLAG=0
00740            XFACT=1.0D0
00750    990     XK=XK-XKD
00760            UBOTM=RHO2/RHO1
00770            IF(XK.GT.(OM/CBOTM)) GO TO 991
00780            IF (XKD.NE.XKDS) GO TO 980
00790            XK=XK+XKD
00800            XKD=XKDS/10
00810            GO TO 990
00820    991     CONTINUE
00830            DO 20 I=1,NMIN
00840            ISC(I)=0
00850    20      MSC(I)=0
00860    994     CONTINUE
00870            UK=DSQRT((XK**2)-((OM/CBOTM)**2))
00880            IF(MFLAG.EQ.1.AND.ICAL.NE.1) JUPA(ICOUNT)=JUP
00890            JTER=JTER+1
00900            XN=H2*(XK**2)
00910            DO 3     J=1,NMIN
00920            JUP=0
00930            TT=T(J)
00940            IF(X(J).GT.XMAX) MFLAG1=0
00950            IF(J.EQ.1) XNEW=-XINCR
00960            SN=S(J)/DABS(S(J))
00970            AL=SN*(S(J)**TH)
00980            B(J)=C(J)=XN
00990            BE=B(J)/S(J)
01000            DX=-AL*(X(J)+BE)
01010            DX1=-AL*(X(J+1)+BE)
01020            SW=1.D0
01030            IF(J.EQ.1) CALL START(AL,BE,X(J),X(J+1),UOUT,UPOUT,
01040        1   XNEW,XINCR,SW,TT)
01050            IF(J.EQ.1) GO TO 812
01060            CONTINUE
01070            IF(J.LE.JD) GO TO 605
01080            IF(KFLAG.EQ.1) GO TO 600
01090            IF(DX1.LE.7) GO TO 605
01100            KFLAG=1
01110            JD=J-1
01120            GO TO 600
01130    605     CONTINUE
01140            IF(DX.GT.DM) CALL LARGE(AL,BE,X(J),X(J+1),UIN,
01150        1   UOUT,UPIN,UPOUT,XNEW,XINCR,TT)
```

(4)

(5)

(6)

(7)

(8)

```
01160            IF(DX.LE.DM) CALL PROP(AL,BE,X(J),X(J+1),UIN,
01170          1 UOUT,UPIN,UPOUT,XNEW,XINCR,TT)
01180    812     UIN=UOUT
01190            UPIN=UPOUT
01200            IF (MFLAG.NE.1.OR.ICAL.EQ.1) GO TO 115
01210            IF(J.NE.1) GO TO 120
01220            CALL DAIRY(-AL*BE,AI,AIP,BI,BIP)
01230            UOUT0=0.
01240            UPOUT0=DABS(BI*AIP-AI*BIP)                          (8)
01250            JPL=1
01260            WRITE(NFILE(7),130) JPL,KFLAG,X(1),AL,BE,UOUT0,UPOUT0
01270    120     CONTINUE
01280                    JPL=J+1
01290            WRITE(NFILE(7),130) JPL,KFLAG,X(J+1),AL,BE,UOUT,UPOUT
01300    130     FORMAT(2I4,5D)
01310    115     CONTINUE
01320    3       CONTINUE
01330            SCALE=UBOTM/UIN
01340            SCALE2=(SCALE**2)
01350            UPBOTM=-H*UK/-AL
01360            FUNC=UIN*UPBOTM-UPIN*UBOTM
01370            XNORM2=0.0D0
01380            GO TO 45
01390    600     CONTINUE
01400            KFLAG=1
01410            ISW=ICOUNT+1
01420            JX=J
01430            DO 4 J=NMIN,JX,-1
01440            JUP=NMIN-JX+1
01450            FCT=1.0D0
01460            MS=0
01470            IF(MFLAG1.EQ.1) MS=1
01480            IF(X(J).GT.XMAX.AND.MS.EQ.1) MFLAG1=0
01490            IF(J.EQ.NMIN) XNEW=1.0D0
01500            SN=S(J)/DABS(S(J))
01510            AL=SN*(S(J)**TH)
01520            TD(J)=ALPH(J+1)/ALPH(J)
01530            IF(J.LT.NMIN) GO TO 606
01540            TD(J)=1.0D0/AL
01550            UDIN=UBOTM*XFACT/1.0D25                             (9)
01560            UPDIN=-H*UK/-AL
01570            UPDIN=UPDIN*XFACT/1.0D25
01580    606     CONTINUE
01590            B(J)=C(J)-XN
01600            BE=B(J)/S(J)
01610            DX1=-AL*(X(J)+BE)
01620            DX=-AL*(X(J+1)+BE)
01630            IF(DX.GT.DM) CALL LG(AL,BE,X(J+1),X(J),UDIN,
01640          1 UDOUT,UPDIN,UPDOUT,XNEW,XINCR,TD(J),XS(J),XT(J))
01650            IF(DX.LE.DM) CALL PR(AL,BE,X(J+1),X(J),UDIN,
01660          1 UDOUT,UPDIN,UPDOUT,XNEW,XINCR,TD(J),XS(J),XT(J))
01670            IF(J.EQ.JX) GO TO 300
01680            IF(MFLAG.EQ.1) GO TO 110
01690            IF(IFLAG.EQ.1) GO TO 608
01700    603     IF(IFLAG.NE.1.AND.DABS(UDOUT).GT.1.D-5)ISC(J)=ISC(J)+1
01710            IF(ISC(J).EQ.0) GO TO 300                           (10)
01720            UDOUT=UDOUT/1.D10
01730            UPDOUT=UPDOUT/1.D10
01740            IF(DABS(UDOUT).GT.1.D-5) GO TO 603
01750            GO TO 300
```

```
01760   608     UDOUT=UDOUT/10.D0**(ISC(J)*10)
01770           UPDOUT=UPDOUT/10.D0**(ISC(J)*10)
01780           GO TO 300
01790   110     CONTINUE
01800           IF(ICAL.EQ.1) GO TO 301
01810   604     IF(DABS(UDOUT).GT.1.D-5)MSC(J)=MSC(J)+1
01820           IF(MSC(J).EQ.0) GO TO 300
01830           UDOUT=UDOUT/1.D10
01840           UPDOUT=UPDOUT/1.D10
01850           IF(DABS(UDOUT).GT.1.D-5) GO TO 604
01860           GO TO 300
01870   301     UDOUT=UDOUT/10.D0**(MSC(J)*10)
01880           UPDOUT=UPDOUT/10.D0**(MSC(J)*10)
01890   300     CONTINUE
01900           JPL=J+1
01910           UX=0
01920           PX=0.
01930           IF(J.EQ.JX.AND.JX.NE.NMIN) UX=UDIN
01940           IF(J.EQ.JX.AND.JX.NE.NMIN) PX=UPDIN
01950           IF(ICAL.NE.1.AND.MFLAG.EQ.1)WRITE(NFILE(7),130)
01960         1 JPL,KFLAG,X(J),AL,BE,UX,PX
01970           UDIN=UDOUT
01980           UPDIN=UPDOUT
01990           IF(MS.EQ.1) MFLAG1=1
02000   4       CONTINUE
02010           UDOUT=UDOUT/1.0D15
02020           UPDOUT=UPDOUT/1.0D15
02030           FUNC=UIN*UPDOUT-UPIN*UDOUT
02040           IF(MFLAG.NE.1)GO TO 45
02050           NS=0
02060           IF(JX.EQ.NMIN) GO TO 22
02070           DO 28 I=JX+1,NMIN
02080   28      NS=NS-MSC(I)+ISC(I)
02090           DO 47 K=JX,NMIN
02100           NSC(ICOUNT,K)=NS
02110           IF(K.LE.JX) GO TO 47
02120           DO 46 N=JX,K
02130   46      NSC(ICOUNT,K)=NSC(ICOUNT,K)+MSC(N)
02140   47      CONTINUE
02150   22      XNORM2=0.D0
02160           XST=XS(JX)+XT(JX)
02170           IF(NS.EQ.0) XNORM2=XST
02180           IF(NS.NE.0.AND.DLOG10(XST)-20.D0*NS.GT.-30.D0)
02190         1 XNORM2=10.D0**(DLOG10(XST)-NS*20.D0)
02200   45      CONTINUE
02210           IF(ICAL.EQ.1) GO TO 892
02220   C       TYPE 1,XK,FUNC
02230           XNOR1=0.0D0
02240           IF(KFLAG.EQ.0) XNOR1=RHO2/(2.0D0*H*UK)
02250           XNR=XNOR1+(XNORM2+XNORM)*SCALE2
02260           IF(XNR.GT.0.0D0) SQN=(1.0D0/DSQRT(XNR))*DABS(SCALE)
02270           IF(MFLAG.EQ.1) WRITE(NTYPE,1) SQN
02280           IF(MFLAG.NE.1) GO TO 165
02290   C       STORE U,UP AT LAYER BOUNDARIES FOR MODE JCOUNT
02300           WRITE(21,170) NUMBER
02310   170     FORMAT(I4)
02320           CLOSE(UNIT=29)
02330           DO 160 IN=1,NUMBER
02340           READ(NFILE(7),130) JC,KC,XC,ALC,BEC,UXC,UPXC
02350           UXC=UXC*SQN
```

(10) (9) (11) ((12)

```
02360                 UPXC=UPXC*SQN
02370                 IF(KC.EQ.1) UXC=UXC*10**(-10*NSC(ICOUNT,JX))
02380                 IF(KC.EQ.1) UPXC=UPXC*10**(-10*NSC(ICOUNT,JX))
02390    160          WRITE(21,132) XC,ALC,BEC,UXC,UPXC
02400    132          FORMAT(5D)
02410    165          CONTINUE
02420                 CLOSE(UNIT=NFILE(7))
02430                 IF(MFLAG.NE.1.OR.NOPT.EQ.1) GO TO 892
02440                 ICAL=1
02450                 XMAX=XMAX2/H
02460                 XINCR=XINCR2/H
02470                 MFLAG1=1
02480                 GO TO 994
02490    892          IF(MFLAG.EQ.1) GO TO 890
02500                 IF(DABS(EDIF).LT.1.0D-15) ITER=1
02510                 IF(ITER.EQ.0) GO TO 444
02520                 ICOUNT=ICOUNT+1
02530                 EDIF=1.0D0
02540                 MFLAG1=1
02550                 MFLAG=1
02560                 IF(KFLAG.EQ.1) XFACT=UIN/(UDOUT*1.0D15)
02570                 WRITE(NTYPE,1) XK
02580                 TYPE 770,XK,ICOUNT
02590                 ITER=0
02600                 GO TO 994
02610    444          CONTINUE
02620                 FU(JTER)=FUNC
02630                 EIG(JTER)=XK
02640                 IF(IFLAG.EQ.1) GO TO 880
02650                 IF(JTER.EQ.1) GO TO 990
02660                 PROD=FU(JTER)/DABS(FU(JTER))
02670                 PROD=PROD*FU(JTER-1)/DABS(FU(JTER-1))
02680                 IF(PROD.GT.0.0D0) GO TO 990
02690                 IFLAG=1
02700                 XK=EIG(JTER)-DSQRT(FU(JTER)/(FU(JTER)-FU(JTER-1)))
02710           1     *(EIG(JTER)-EIG(JTER-1))
02720                 GO TO 994
02730    880          JMIN=JTER-1
02740                 K=JMIN
02750                 GO TO 882
02760                 DO 881 K=JMIN,1,-1
02770                 PROD=FU(JTER)/DABS(FU(JTER))
02780                 PROD=PROD*FU(K)/DABS(FU(K))
02790                 IF(PROD.LT.0.0D0) GO TO 882
02800    881          CONTINUE
02810    882          SE=(FU(JTER)-FU(K))/(EIG(JTER)-EIG(K))
02820                 SIN=FU(JTER)-(SE*EIG(JTER))
02830                 XK=-SIN/SE
02840                 EDIF=XK-EIG(JTER)
02850    770          FORMAT(1D,I5)
02860                 GO TO 994
12  02870    980      CONTINUE
11  02880             TYPE 31,(JUPA(I),I=1,ICOUNT)
10  02890    31       FORMAT('      UP-LAYERS #'/1014)
9   02900             CLOSE (UNIT=NTYPE)
8   02910             NTT=0
7   02920             DO 41 I=1,NMIN
6   02930    41       NTT=NTT+NPPL(I)
5   02940             DO 16 I=1,ICOUNT
4   02950             IF(I.GE.ISW.OR.NOPT.EQ.1) IRLAY=IRLAY1
3
```

(13)

(14)

(15)

(16)

(17)

```
02960                IREC=NTT-IREC1+1
02970                DO 42 J=1,NMIN-JUPA(I)
02980    42          IREC=IREC+NPPL(J)
02990    95          IF(I.GE.ISW) IREC=IREC1
03000                IRECA(I)=IREC
03010                READ(NTYPE,1) XKA(IPR,I)
03020                DO 70 K=1,ISLAY
03030                READ(NTYPE,2) BXTMP
03040                IF(K.EQ.ISOU) BX=BXTMP
03050    70          IF(K.EQ.IREC.AND.NOPT.EQ.1) CX=BXTMP
03060                READ(NTYPE,1) XN
03070                IF(NOPT.EQ.1) GO TO 30
03080                DO 71 K=1,IRLAY
03090                READ(NTYPE,2) CXTMP
03100    71          IF(K.EQ.IREC) CX=CXTMP
03110    30          CONTINUE
03120                NSPPL=0
03130                DO 39 L=1,NMIN
03140                NSPPL=NSPPL+NPPL(L)
03150                KL=L
03160                IF(NSPPL.GT.IREC1) GO TO 40
03170    39          CONTINUE
03180    40          CONTINUE
03190                UO(IPR,I)=BX*XN
03200    16          UF(IPR,I)=CX*XN*10.D0**(-10*NSC(I,KL))
03210                CLOSE (UNIT=NTYPE)
03220    19          XKD=XKDS
03230                TYPE 48,(IRECA(I),I=1,ICOUNT)
03240    48          FORMAT('    # OF REC IN WFCT FILE ='/10I4)
03250                TYPE 18,IPR,ICOUNT,ISW
03260    18          FORMAT('   END OF PROFILE #',I2,'     # OF EV IN THIS P
03270           1ROFILE =',I2,'    ISW=',I2)
03280    15          CONTINUE
03290                DO 17 J=IPRMIN,IPRMAX
03300                WRITE(NFILE(3),7) (UO(J,I),I=1,IM)
03310                WRITE(NFILE(4),7) (UF(J,I),I=1,IM)
03320                WRITE(NFILE(5),7) (XKA(J,I),I=1,IM)
03330    17          CONTINUE
03340    2           FORMAT(D20.8)
03350    7           FORMAT(8D)
03360    996         STOP
03370                END
```

(17)

```
00010            SUBROUTINE SELAUT(IP,NFIL,NTEMP)
00020            DOUBLE PRECISION X(55),Y(55)
00030            DO 10  I=1,IP
00040            READ(NFIL,2) N
00050            READ(NFIL,1) (Y(J),J=1,N+1)
00060      10    READ(NFIL,1) (X(J),J=1,N+1)
00070            WRITE(NTEMP,2) N
00080            DO 20 I=1,N
00090      20    WRITE(NTEMP,3) X(I),Y(I)
00100            WRITE(NTEMP,3) Y(N+1)
00110      1     FORMAT(5D14.7)
00120      2     FORMAT(I3)
00130      3     FORMAT(2D)
00140            RETURN
00150            END
```

```
00010            SUBROUTINE PARAM
00020            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
00030            DIMENSION NPPL(60),XKST(10),NFILE(7)
00040            COMMON /DATCOM/XKST,FREQ,RHO1,RHO2,XKD,XMAX1,XMAX2,
00050          1 XINCR1,XINCR2,NPPL,JMIND,NFILE,ISLAY,IRLAY,
00060          2 ISOU,IREC,NOPT,IPRMIN,IPRMAX,ICT,IRLAY1,IREC1
00070            DATA NPPL/10,14,24,57*0/
00080            DATA XKST/10*1000.00/
00090            DATA NFILE/38,24,25,27,28,21,29/
00100            CON=.30487804
00110            FREQ=5.D0
00120            RHO1=1.D0
00130            RHO2=1.D0
00140            XMAX1=100.
00150            XINCR1=50.
00160            XMAX2=100.
00170            ISLAY=10
00180            IRLAY=51
00190            ISOU=1
00200            IREC=2
00210            IRLAY1=51
00220            IREC1=2
00230            XINCR2=100.
00240            XKD=.4D-4
00250            ICT=20
00260            JMIND=0
00270            IPRMIN=1
00280            IPRMAX=4
00290            NOPT=1
00300            RETURN
00310            END
```

AD-A062 915 CATHOLIC UNIV OF AMERICA WASHINGTON D C DEPT OF PHYSICS F/G 9/2
A FORTRAN CODE FOR THE CALCULATION OF SOUND PROPAGATION IN A RA--ETC(U)
JUN 78 A NAGL, G L ZARUR, H UEBERALL N00173-77-C-0008

UNCLASSIFIED NL

2 OF 2
AD
AO 6291 5

END
DATE
FILMED
3 --79
DDC

```
00010          SUBROUTINE DAIRY(DX,AI,AIP,BI,BIP)                          26230
00020            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
00030      C  FOR DOUBLE PRECISION ARGUMENTS, THIS ROUTINE CALCULATES THE AIRY  26240
00040      C     FUNCTION AI(X) AND ITS DERIVATIVE AIP(X).  IT ALSO FINDS     26250
00050      C     THE OTHER REAL LINEARLY INDEPENDENT SOLUTION BI(X) AND       26260
00060      C     ITS DERIVATIVE BIP(X).                                       26270
00070      C     THE DEFINITIONS AND NORMALIZATIONS ARE AS IN NBS HANDBOOK     26280
00080      C     OF MATHEMATICAL FUNCTIONS,P,446                              26290
00090      C     THE METHODS USED ARE POWER SERIES EXPANSION FOR SMALL X      26300
00100      C     AND GAUSSIAN INTEGRATION FOR LARGE X                         26310
00110          DIMENSION X(16),W(16),XSQ(16)                                26320
00120      C     DOUBLE PRECISION DX,AI,AIP,BI,BIP                            26330
00130      C     DOUBLE PRECISION    XS ,XCUBE,AISUM,AIPSUM                   26340
00140      C     DOUBLE PRECISION DF,DFP,DG,DGP                               26350
00150      C     DOUBLE PRECISION FJM2,FJM1,FJ,FJP1,FJP2,FACTOR               26360
00160      C     DOUBLE PRECISION C1,C2,ROOT3                                 26370
00170      C     DOUBLE PRECISION DZETA,DARG,DROOTX                           26380
00180      C     DOUBLE PRECISION ROOT4X,S,CO,RATIO,EFAC,ZETASQ               26390
00190      C     DOUBLE PRECISION SUMR,SUMI,SUMRP,SUMIP,TERMR,TERMI           26400
00200      C     DOUBLE PRECISION DZERO,DA,DB,DEN,ONE                         26410
00210      C     DOUBLE PRECISION X,W,XSQ                                     26420
00220      C     DOUBLE PRECISION RSQ,TEMP,    RTPI,RTPI2                     26430
00230      C     DOUBLE PRECISION TERMA,TERMB                                 26440
00240          LOGICAL NEEDBI                                               26450
00250          DATA DZERO,ONE /0.0D0,1.0D0/                                 26460
00260          DATA ROOT3/1.732050807568877D0/                              26470
00270          DATA C1,C2 /.355028053887817D0, .258819403792807D0/          26480
00280          DATA RTPI /.282094791773878 1D0/                             26490
00290          DATA RTPI2/.564189583547756 2D0/                             26500
00300      C  POSITIONS AND WEIGHTS FOR 10-TERM SUM FOR AIRY FUNCTIONS      26510
00310          DATA W( 1) /  3.1542515762964787D-14/                        26520
00320          DATA W( 2) /  6.6394210819584921D-11/                        26530
00330          DATA W( 3) /  1.7583889061345669D-08/                        26540
00340          DATA W( 4) /  1.3712392370435815D-06/                        26550
00350          DATA W( 5) /  4.4350966392843500-05/                         26560
00360          DATA W( 6) /  7.1555010917718255D-04/                        26570
00370          DATA W( 7) /  6.4889566103335381D-03/                        26580
00380          DATA W( 8) /  3.6440415875773282D-02/                        26590
00390          DATA W( 9) /  1.4399792418590999D-01/                        26600
00400          DATA W(10) /  8.1231141336261486D-01/                        26610
00410          DATA X( 1) /  1.4083081072180964D+01/                        26620
00420          DATA X( 2) /  1.0214885479197331D+01/                        26630
00430          DATA X( 3) /  7.4416018450450930D+00/                        26640
00440          DATA X( 4) /  5.3070943061781927D+00/                        26650
00450          DATA X( 5) /  3.6340135029132462D+00/                        26660
00460          DATA X( 6) /  2.3310652303052450D+00/                        26670
00470          DATA X( 7) /  1.3447970824609268D+00/                        26680
00480          DATA X( 8) /  6.4188858369567296D-01/                        26690
00490          DATA X( 9) /  2.0100345998121046D-01/                        26700
00500          DATA X(10) /  8.0594359172052833D-03/                        26710
00510          DATA XSQ( 1) /0.19833317248562170D 03/                       26720
00520          DATA XSQ( 2) /0.10434388535311650D 03/                       26730
00530          DATA XSQ( 3) /0.55377438020178170D 02/                       26740
00540          DATA XSQ( 4) /0.28165249974668990D 02/                       26750
00550          DATA XSQ( 5) /0.13206054139355800D 02/                       26760
00560          DATA XSQ( 6) /0.54338651079380440D 01/                       26770
00570          DATA XSQ( 7) /0.18084791929954200D 01/                       26780
00580          DATA XSQ( 8) /0.41202095387883690D 00/                       26790
00590          DATA XSQ( 9) /0.40402390924418070D-01/                       26800
00600          DATA XSQ(10) /0.64954507303538390D-04/                       26810
```

```
00610   C  POSITIONS AND WEIGHTS FOR  4-TERM SUM FOR AIRY FUNCTIONS                26820
00620         DATA W(11) /  4.77639030575772630-05/                                26830
00630         DATA W(12) /  4.99143064329109590-03/                                26840
00640         DATA W(13) /  8.61698469938403120-02/                                26850
00650         DATA W(14) /  9.08790958459811020-01/                                26860
00660         DATA X(11) /  3.91983295544550910+00/                                26870
00670         DATA X(12) /  1.69156190048235040+00/                                26880
00680         DATA X(13) /  5.02755324672630180-01/                                26890
00690         DATA X(14) /  1.92470605620156920-02/                                26900
00700        DATA XSQ(11) /0.153650903985966700 02/                                26910
00710        DATA XSQ(12) /0.286138166316346100 01/                                26920
00720        DATA XSQ(13) /0.252762916486681800 00/                                26930
00730        DATA XSQ(14) /0.370449340277899800-03/                                26940
00740   C  POSITIONS AND WEIGHTS FOR  2-TERM SUM FOR AIRY FUNCTIONS                26950
00750         DATA W(15) /  9.68072805957736040-01/                                26960
00760         DATA W(16) /  3.19271940421639580-02/                                26970
00770         DATA X(15) /  3.68006018661530440-02/                                26980
00780         DATA X(16) /  1.05924693821123780+00/                                26990
00790        DATA XSQ(15) /0.135428429771110700-02/                                27000
00800        DATA XSQ(16) /0.112200407610988100 01/                                27010
00802          IF(DX.GT.-1000.D0) GO TO 991
00804          CALL SINCOS(DX,AI,AIP,BI,BIP)
00806          RETURN
00808   991    CONTINUE
00810        IF(DX.LT.-5.0D0) GO TO 100                                            27020
00820        NEEDBI=.FALSE.                                                        27030
00830        IF(DX.GT.3.700) GO TO 200                                             27040
00840   C    THIS ROUTE FOR SMALLX, USING POWER SERIES.                            27050
00850   C    INITIALIZE                                                            27060
00860   10   XS  = DX*DX                                                          27070
00870        XCUBE = XS *DX                                                        27080
00880        XS  = XS *0.500                                                       27090
00890        DF = C1                                                               27100
00900        DFP = C1*XS                                                           27110
00910        DG = C2*DX                                                            27120
00920        DGP = C2                                                              27130
00930        AISUM = DF - DG                                                       27140
00940        AIPSUM = DFP - DGP                                                    27150
00950        BI = DF + DG                                                          27160
00960        BIP = DFP + DGP                                                       27170
00970        FJM2=-2.0D0                                                           27180
00980   20   FJM2=FJM2+3.0D0                                                       27190
00990        FJM1=FJM2+ONE                                                         27200
01000        FJ=FJM1+ONE                                                           27210
01010        FJP1=FJ+ONE                                                           27220
01020        FJP2=FJP1+ONE                                                         27230
01030        RATIO = XCUBE/FJ                                                      27240
01040        DF = DF*RATIO/FJM1                                                    27250
01050        DFP = DFP*RATIO/FJP2                                                  27260
01060        DG = DG*RATIO/FJP1                                                    27270
01070        DGP = DGP*RATIO/FJM2                                                  27280
01080        BI = BI + (DF+DG)                                                     27290
01090        BIP = BIP + (DFP+DGP)                                                 27300
01100        IF(NEEDBI) GO TO 80                                                   27310
01110        AISUM = AISUM + (DF-DG)                                               27320
01120        AIPSUM = AIPSUM + (DFP-DGP)                                           27330
01130   C    CONVERGENCE TEST                                                      27340
01140   80   IF(DABS(DF).GT.1.0D-16) GO TO 20                                      27350
01150   C    CONVERGENCE, COMPUTE FUNCTIONS                                        27360
01160   99   BI = ROOT3*BI                                                         27370
```

```
01170          BIP = ROOT3*BIP                                          27380
01180    C  THIS RETURNS IF X IS BETWEEN 3.7 AND 8.0, SINCE IN SUCH CASES MORE   27390
01190    C  ACCURATE VALUES OF AI AND AIP HAVE ALREADY BEEN FOUND BY GAUSSIAN    27400
01200    C  INTEGRATION                                                  27410
01210          IF(NEEDBI)RETURN                                         27420
01220          AI = AISUM                                               27430
01230          AIP = AIPSUM                                             27440
01240          RETURN                                                   27450
01250    C  GAUSSIAN INTEGRATION FOR LARGE NEGATIVE X                   27460
01260    100   DROOTX = DSQRT(-DX)                                      27470
01270          ROOT4X = DSQRT(DROOTX)                                   27480
01280          DZETA = -.6666666666666667*DX*DROOTX                     27490
01290          DARG = DZETA - .7853981633974483                         27500
01300          SUMR = DZERO                                             27510
01310          SUMI = DZERO                                             27520
01320          SUMRP = DZERO                                            27530
01330          SUMIP = DZERO                                            27540
01340    C  TEST TO SEE HOW MANY TERMS ARE NEEDED IN GAUSSIAN INTEGRATION 27550
01350          IF(DX.LT.(-200.D0)) GO TO 140                            27560
01360          IF(DX.LT.(-15.D0)) GO TO 130                             27570
01370    C  THIS CASE FOR DX BETWEEN -5.0 AND -15.0                     27580
01380          LIMLO=1                                                  27590
01390          LIMHI=10                                                 27600
01400          GO TO 149                                                27610
01410    C  THIS CASE FOR DX BETWEEN -15.0 AND -200.                    27620
01420    130   LIMLO=11                                                 27630
01430          LIMHI=14                                                 27640
01440          GO TO 149                                                27650
01450    C  THIS CASE FOR DX.LT.-200.                                   27660
01460    140   LIMLO=15                                                 27670
01470          LIMHI=16                                                 27680
01480    149   ZETASQ=DZETA**2                                          27690
01490          DO 150 K=LIMLO,LIMHI                                     27700
01500          TERMR=W(K)/((ZETASQ+XSQ(K))**2)                          27710
01510          SUMR = SUMR + TERMR                                      27720
01520          TERMR=TERMR*X(K)                                         27730
01530          SUMI=SUMI+TERMR                                          27740
01540          TERMR=TERMR*X(K)                                         27750
01550          SUMRP=SUMRP+TERMR                                        27760
01560    150   SUMIP=SUMIP+TERMR*X(K)                                   27770
01570          SUMR=(SUMR*ZETASQ+SUMRP)*ZETASQ                          27780
01580          TEMP=SUMI*ZETASQ                                         27790
01590          SUMI=(TEMP+SUMIP)*DZETA                                  27800
01600          SUMRP=SUMRP*DZETA                                        27810
01610          SUMIP=SUMIP-TEMP                                         27820
01620    C  FORM AIRY FUNCTIONS                                         27830
01630    196   S = DSIN(DARG)                                           27840
01640          CO = DCOS(DARG)                                          27850
01650          RATIO = RTPI2/ROOT4X                                     27860
01660          AI = RATIO*(CO*SUMR + S*SUMI)                            27870
01670          BI = RATIO*(CO*SUMI - S*SUMR)                            27880
01680          SUMRP=SUMRP+SUMRP                                        27890
01690          RATIO = -.2500/DX                                        27900
01700          FACTOR = -RTPI2*ROOT4X                                   27910
01710          AIP = RATIO*AI - DROOTX*BI + FACTOR*(CO*SUMRP+S*SUMIP)   27920
01720          BIP = RATIO*BI + DROOTX*AI + FACTOR*(CO*SUMIP-S*SUMRP)   27930
01730          RETURN                                                   27940
01740    C   GAUSSIAN INTEGRATION FOR LARGE POSITIVE X                  27950
01750    200   DROOTX = DSQRT(DX)                                       27960
01760          DZETA = .6666666666666667*DX*DROOTX                      27970
```

```
01770          EFAC = DEXP(-DZETA)                                          27980
01780          ROOT4X = DSQRT(DROOTX)                                       27990
01790          AI = DZERO                                                   28000
01800          BI = DZERO                                                   28010
01810          AIP = DZERO                                                  28020
01820          BIP = DZERO                                                  28030
01830          IF(DX.LT.8.0D0) NEEDBI=.TRUE.                                28040
01840    C  TEST TO SEE HOW MANY TERMS ARE NEEDED IN GAUSSIAN INTEGRATION   28050
01850          IF(DX.GT.15.0D0) GO TO 230                                   28060
01860    C  THIS CASE FOR DX BETWEEN 3.7 AND 15.                            28070
01870          LIMLO=1                                                      28080
01880          LIMHI=10                                                     28090
01890          GO TO 249                                                    28100
01900    C  THIS CASE FOR DX GREATER THAN 15.                               28110
01910    230   LIMLO=11                                                     28120
01920          LIMHI=14                                                     28130
01930    249   DO 250 K=LIMLO,LIMHI                                         28140
01940          DA=DZETA+X(K)                                                28150
01950          TERMA = W(K)/DA                                              28160
01960          AI = AI + TERMA                                              28170
01970          AIP=AIP+TERMA*X(K)/DA                                        28180
01980          IF(NEEDBI) GO TO 250                                         28190
01990          DB=DZETA-X(K)                                                28200
02000          TERMB = W(K)/DB                                              28210
02010          BI = BI + TERMB                                              28220
02020          BIP=BIP+TERMB*X(K)/DB                                        28230
02030    250   CONTINUE                                                     28240
02040    C  FORM FUNCTIONS                                                  28250
02050          FACTOR=RTPI*DZETA/ROOT4X                                     28260
02060          RATIO = 0.25D0/DX                                            28270
02070          AI=AI*EFAC*FACTOR                                            28280
02080          AIP=-(DROOTX+RATIO)*AI+RTPI*ROOT4X*EFAC*AIP                  28290
02090    C  THIS IS SATISFIED ONLY FOR X BETWEEN 3.7 AND 8.0  IN THESE CASES 28300
02100    C  THE BI AND BIP ABOUT TO BE COMPUTED ARE NOT SUFFICIENTLY ACCURATE, 28310
02110    C  THUS RETURN TO POWER SERIES FOR BI AND BIP.                     28320
02120          IF(NEEDBI) GO TO 10                                          28330
02130          FACTOR=FACTOR+FACTOR                                         28340
02140          BI=BI*FACTOR/EFAC                                            28350
02150          BIP=(DROOTX-RATIO)*BI-RTPI2*ROOT4X*BIP/EFAC                  28360
02160          RETURN                                                       28370
02170          END                                                          28380
```

```
00100           SUBROUTINE SINCOS(Z,AI,AIP,BI,BIP)
00200           IMPLICIT DOUBLE PRECISION (A-H,O-Z)
00300           PI=3.1415926359000
00400           AR=2.*(-Z)**(1.5)/3.
00500           ARG=AR+PI/4
00600           Z1=(-Z)**(-0.25)
00700           Z2=1./Z1
00800           PI2=1./DSQRT(PI)
00900           AI=PI2*Z1*(DSIN(ARG)-5./72./AR*DCOS(ARG))
01000     20    FORMAT(2D)
01100           BI=PI2*Z1*(DCOS(ARG)+5./72./AR*DSIN(ARG))
01200           AIP=-PI2*Z2*(DCOS(ARG)+7./72./AR*DSIN(ARG))
01300           BIP=PI2*Z2*(DSIN(ARG)-7./72./AR*DCOS(ARG))
01400           RETURN
01500           END
```

```
00010            SUBROUTINE EXTREM(X,A,AP,B,BP,ZETA)
00020            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
00030            DIMENSION XI(4),W(4)
00032            IF (X.GT.15.D0) GO TO 100
00034            CALL DAIRY(X,A,AP,B,BP)
00036            ZETA=1.D0
00038            RETURN
00039    100     CONTINUE
00040            XI(1)=3.91983295544550910D0
00050            XI(2)=1.69156190048235040D0
00060            XI(3)=5.02755324672630180-01
00070            XI(4)=1.92470605620156920-02
00080            W(1)=4.77639030575772630-05
00090            W(2)=4.99143064329109590-03
00100            W(3)=8.61698469938403120-02
00110            W(4)=9.08790958459811020-01
00120            DROOTX=DSQRT(X)
00130            ZETA=0.66666666666667D0*X*DROOTX
00140            ROOT4X=DSQRT(DROOTX)
00150            A=0.0D0
00160            AP=0.0D0
00170            B=0.0D0
00180            BP=0.0D0
00190            DO 1 K=1,4
00200            DA=ZETA+XI(K)
00210            TERMA=W(K)/DA
00220            A=A+TERMA
00230            AP=AP+TERMA*XI(K)/DA
00240            DB=ZETA-XI(K)
00250            TERMB=W(K)/DB
00260            B=B+TERMB
00270            BP=BP+TERMB*XI(K)/DB
00280    1       CONTINUE
00290            RTPI=0.28209479177387810D0
00300            RTPI2=0.56418958354775620D0
00310            FAC=RTPI*ZETA/ROOT4X
00320            RAT=0.250D0/X
00330            A=A*FAC
00340            AP=-(DROOTX+RAT)*A+RTPI*ROOT4X*AP
00350            FAC=FAC*FAC
00360            B=B*FAC
00370            BP=(DROOTX-RAT)*B-RTPI2*ROOT4X*BP
00380            RETURN
00390            END
```

```
      00010            SUBROUTINE PROP(AL,BE,X,XJ,U,UJ,UP,UPJ,XNEW,XINCR,TT)
      00020            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      00030            COMMON /IFLAGS/ MFLAG,MFLAG1,NTYPE
      00040            COMMON /SQNORM/ XNORM
      00050            DM=11.0D0
      00070            DX=-AL*(X+BE)
      00080            DX1=-AL*(XJ+BE)
      00090            CALL DAIRY(DX,AI,AIP,BI,BIP)
      00100            PI=3.14159265358978323840D0
      00110            AJ=PI*((BIP*U)-(TT*BI*UP))
      00120            BJ=PI*((TT*AI*UP)-(AIP*U))
      00125            TT=1.0D0
      00130            IF(MFLAG.NE.1) GO TO 5
      00140            B1=(AJ**2)*(((DX/AL)*(AI**2))-((AIP**2)/AL))
      00150            B2=(BJ**2)*(((DX/AL)*(BI**2))-((BIP**2)/AL))
      00160            B3=(2.0D0*AJ*BJ)*(((DX/AL)*(AI*BI))-((AIP*BIP)/AL))
      00170            BEG=-B1-B2-B3
      00180      5     CONTINUE
      00190            IF(MFLAG1.EQ.1) GO TO 4
      00200            IF((DX.LE.DM).AND.(DX1.LE.DM)) GO TO 3
      00210      4     CONTINUE
      00220      1     XNEW=XNEW+XINCR
      00230            IF(XNEW.GT.XJ) GO TO 2
      00240            DX=-AL*(XNEW+BE)
      00250            IF(DX.LE.DM.OR.IA.EQ.0) GO TO 6
      00255            IF(XNEW-XINCR.LE.X)GO TO 6
      00260            XNEW=XNEW-XINCR
      00270            IF(MFLAG.NE.1) GO TO 7
      00280            DX=-AL*(XNEW+BE)
1413  00290            C1=(AJ**2)*(((DX/AL)*(AI**2))-((AIP**2)/AL))
      00300            C2=(BJ**2)*(((DX/AL)*(BI**2))-((BIP**2)/AL))
      00310            C3=(2.0D0*AJ*BJ)*(((DX/AL)*(AI*BI))-((AIP*BIP)/AL))
      00320            CEN=C1+C2+C3
      00330            ZNORM=-CEN-BEG
      00340            XNORM=XNORM+ZNORM
      00350      7     CONTINUE
      00370            CALL LARGE(AL,BE,XNEW,XJ,WF,UJ,WP,UPJ,XNEW,XINCR,TT)
      00380            RETURN
      00390      6     CONTINUE
      00400            CALL DAIRY(DX,AI,AIP,BI,BIP)
      00410            IA=IA+1
      00420            WF=AJ*AI+BJ*BI
      00430            WP=AJ*AIP+BJ*BIP
      00440            IF(MFLAG1.EQ.1) WRITE(NTYPE,11) WF
      00450      11    FORMAT(2D20.8)
      00460            GO TO 1
      00470      2     XNEW=XNEW-XINCR
      00480      8     DX=-AL*(XJ+BE)
      00482      120   CONTINUE
      00484            IF(XNEW+XINCR.GT.XJ) GO TO 150
      00486            XNEW = XNEW+XINCR
   12 00486            GO TO 120
   11 00488      150   XNEW=XNEW-XINCR
   10 00490            CALL DAIRY(DX,AI,AIP,BI,BIP)
    9 00500            UJ=AJ*AI+BJ*BI
    8 00510            UPJ=AJ*AIP+BJ*BIP
    7 00520            IF(MFLAG.NE.1) RETURN
    6 00530            C1=(AJ**2)*(((DX/AL)*(AI**2))-((AIP**2)/AL))
    5 00540            C2=(BJ**2)*(((DX/AL)*(BI**2))-((BIP**2)/AL))
    4 00550            C3=(2.0D0*AJ*BJ)*(((DX/AL)*(AI*BI))-((AIP*BIP)/AL))
    3
```



```
      00560            CEN=C1+C2+C3
      00570      100   FORMAT(7D14.3)
      00580            ZNORM=-CEN-BEG
      00590            XNORM=XNORM+ZNORM
      00600            RETURN
      00610            END
```

```
      SUBROUTINE PR(AL,BE,X,XJ,U,UJ,UP,UPJ,XNEW,XINCR,TT,
     1 X1SQ,X2SQ)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /IFLAGS/ MFLAG,MFLAG1,NTYPE
      COMMON /JC/ JCOUNT
      X1SQ=0.0D0
      X2SQ=0.0D0
      DM=11.0D0
      IA=0
      DX=-AL*(X+BE)
      DX1=-AL*(XJ+BE)
      CALL DAIRY(DX,AI,AIP,BI,BIP)
      PI=3.14159265358979323840D0
      AJ=PI*((BIP*U)-(TT*BI*UP))
      BJ=PI*((TT*AI*UP)-(AIP*U))
      TT=1.0D0
      IF(MFLAG.NE.1) GO TO 5
      B1=(AJ**2)*(((DX/AL)*(AI**2))-((AIP**2)/AL))
      B2=(BJ**2)*(((DX/AL)*(BI**2))-((BIP**2)/AL))
      B3=(2.0D0*AJ*BJ)*(((DX/AL)*(AI*BI))-((AIP*BIP)/AL))
      BEG=-B1-B2-B3
    5 CONTINUE
      IF(MFLAG1.EQ.1) GO TO 4
      IF((DX.LE.DM).AND.(DX1.LE.DM)) GO TO 3
    4 CONTINUE
    1 XNEW=XNEW-XINCR
      IF(XNEW.LT.XJ) GO TO 2
      DX=-AL*(XNEW+BE)
      IF(DX.LE.DM) GO TO 6
      XNEW=XNEW+XINCR
      IF(MFLAG.NE.1) GO TO 7
      DX=-AL*(XNEW+BE)
      C1=(AJ**2)*(((DX/AL)*(AI**2))-((AIP**2)/AL))
      C2=(BJ**2)*(((DX/AL)*(BI**2))-((BIP**2)/AL))
      C3=(2.0D0*AJ*BJ)*(((DX/AL)*(AI*BI))-((AIP*BIP)/AL))
      CEN=C1+C2+C3
      X1SQ=CEN+BEG
    7 CONTINUE
  375 FORMAT(' PR',5D15.3)
      CALL LG(AL,BE,XNEW,XJ,WF,UJ,WP,UPJ,XNEW,XINCR,TT,
     1 X2SQ,Y2SQ)
      RETURN
    6 CONTINUE
      CALL DAIRY(DX,AI,AIP,BI,BIP)
      WF=AJ*AI+BJ*BI
      WP=AJ*AIP+BJ*BIP
      IF(MFLAG1.EQ.1) WRITE(NTYPE,11) WF
   11 FORMAT(1D20.8)
      GO TO 1
    2 XNEW=XNEW+XINCR
    3 DX=-AL*(XJ+BE)
   12 CALL DAIRY(DX,AI,AIP,BI,BIP)
   11 UJ=AJ*AI+BJ*BI
   10 UPJ=AJ*AIP+BJ*BIP
    9 IF(MFLAG.NE.1) RETURN
    8 C1=(AJ**2)*(((DX/AL)*(AI**2))-((AIP**2)/AL))
    7 C2=(BJ**2)*(((DX/AL)*(BI**2))-((BIP**2)/AL))
    6 C3=(2.0D0*AJ*BJ)*(((DX/AL)*(AI*BI))-((AIP*BIP)/AL))
    5 CEN=C1+C2+C3
    4 X1SQ=CEN+BEG
    3

      RETURN
      END
```

```
00010            SUBROUTINE START(AL,BE,X,XJ,U,UP,XNEW,XINCR,SW,TT)
00020            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
00030            COMMON /IFLAGS/ MFLAG,MFLAG1,NTYPE
00040            COMMON /SQNORM/ XNORM
00050            DX=-AL*(X+BE)
00060            DX1=-AL*(XJ+BE)
00070            IF((DX.GT.15.0D0).OR.(DX1.GT.15.0D0)) GO TO 4
00080            CALL DAIRY(DX,AI,AIP,BI,BIP)
00090            A=SW*BI*TT
00100            B=-AI*SW*TT
00110            AJ=A
00120            BJ=B
00130            IF(MFLAG.NE.1) GO TO 5
00140            B1=(AJ**2)*(((DX/AL)*(AI**2))-((AIP**2)/AL))
00150            B2=(BJ**2)*(((DX/AL)*(BI**2))-((BIP**2)/AL))
00160            B3=(2.0D0*AJ*BJ)*(((DX/AL)*(AI*BI))-((AIP*BIP)/AL))
00170            BEG=-B1-B2-B3
00180     5      CONTINUE
00190            DX=-AL*(XJ+BE)
00200            CALL DAIRY(DX,AI,AIP,BI,BIP)
00210            U=A*AI+B*BI
00220            UP=A*AIP+B*BIP
00230            IF(MFLAG.NE.1) RETURN
00240            ZA=AJ**2
00250            ZB=(DX/AL)
00260            ZC=AI**2
00270            ZD=(AIP**2)/AL
00280            C1=ZA*((ZB*ZC)-ZD)
00290            C1=(AJ**2)*(((DX/AL)*(AI**2))-((AIP**2)/AL))
00300            C2=(BJ**2)*(((DX/AL)*(BI**2))-((BIP**2)/AL))
00310            C3=(2.0D0*AJ*BJ)*(((DX/AL)*(AI*BI))-((AIP*BIP)/AL))
00320            CEN=C1+C2+C3
00330            ZNORM=-CEN-BEG
00340            XNORM=XNORM+ZNORM
00350     1      XNEW=XNEW+XINCR
00360            IF(XNEW.GT.XJ) GO TO 3
00370            DX=-AL*(XNEW+BE)
00380            CALL DAIRY(DX,AI,AIP,BI,BIP)
00390            WF=A*AI+B*BI
00400            IF((MFLAG1.EQ.1).AND.(XNEW.GT.0.0D0)) WRITE(NTYPE,11)
00410          1 WF
00420            WP=A*AIP+B*BIP
00430     11     FORMAT(2D20.8)
00440            GO TO 1
00450     3      XNEW=XNEW-XINCR
00460            RETURN
00470     4      CONTINUE
00480     C      DX=-AL*(X+BE)
00490     C      CALL EXTREM(DX,AZ,AZP,BZ,BZP,Z)
00500            XNEW=X
00510            T=1.0D0
12   00520     C      A=BZ*SW
11   00530     C      B=-AZ*SW
10   00540     C      U1=A*AZ+B*BZ
9    00550     C      UP1=A*AZP+B*BZP
8    00555            U1=0
7    00556            UP1=-TT/3.141592653589783200
6    00555     C      TYPE 11,X,DX,U1,UP1
6    00580            CALL LARGE(AL,BE,X,XJ,U1,U,UP1,UP,XNEW,XINCR,T)
4    00590            RETURN
3
00600            END
```

```
00010            SUBROUTINE LARGE(AL,BE,X,XJ,U,UJ,UP,UPJ,XNEW,XINCR,TT)
00020            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
00030            COMMON /IFLAGS/ MFLAG,MFLAG1,NTYPE
00040            COMMON /SQNORM/ XNORM
00050            DM=11.0D0
00060            UINIT=U
00070            SUMO=0.0D0
00080            SUME=0.0D0
00090            IA=0
00100            PI=3.14159265358979323840D0
00110            DX=-AL*(X+BE)
00120            CALL EXTREM(DX,AZ,AZP,BZ,BZP,Z)
00130    1       XNEW=XNEW+XINCR
00135            IF(XNEW.LT.X) GO TO 1
00140            IF(XNEW.GT.XJ) GO TO 2
00150            DX=-AL*(XNEW+BE)
00160            IF(DX.GT.DM) GO TO 4
00170            XNEW=XNEW-XINCR
00180            IF(MFLAG.NE.1) GO TO 5
00190            IF(IDF.EQ.0) SUME=SUME-(4.0D0*U1*U1)
00200            IF(IDF.NE.0) SUMO=SUMO-(2.0D0*U1*U1)
00210            SEND=(UINIT**2)+(U1**2)
00220            ZNORM=(XINCR/3.0D0)*(SUME+SUMO+SEND)
00230            XNORM=XNORM+ZNORM
00240    5       CONTINUE
00250            CALL PROP(AL,BE,XNEW,XJ,U,UJ,UP,UPJ,XNEW,XINCR,TT)
00260            RETURN
00270    4       CONTINUE
00280            CALL EXTREM(DX,AZ2,AZP2,BZ2,BZP2,Z2)
00290            DEX=DEXP(Z-Z2)
00300            C11=PI*((AZ2*BZP*DEX)-((BZ2*AZP)/DEX))
00310            C12=(PI*TT)*(((BZ2*AZ)/DEX)-(AZ2*BZ*DEX))
00320            C21=(PI/TT)*((AZP2*BZP*DEX)-((BZP2*AZP)/DEX))
00330            C22=PI*(((BZP2*AZ)/DEX)-(AZP2*BZ*DEX))
00340            U1=C11*U+C12*UP
00350            IF(MFLAG1.EQ.1) WRITE(NTYPE,11) U1
00360    11      FORMAT(2D20.8)
00370            IF(MFLAG.NE.1) GO TO 3
00380            IA=IA+1
00390            IH=IA/2
00400            IDF=(IA-2*IH)
00410            IF(IDF.EQ.0) SUME=SUME+(4.0D0*U1*U1)
00420            IF(IDF.NE.0) SUMO=SUMO+(2.0D0*U1*U1)
00430    3       CONTINUE
00440            U1P=C21*U+C22*UP
00450            U1P=U1P*TT
00460            TT=1.0D0
00470            DX=-AL*(XNEW+BE)
00480            AZ=AZ2
00490            AZP=AZP2
00500            BZ=BZ2
00510            BZP=BZP2
00520            Z=Z2
00530            U=U1
00540            UP=U1P
00550    100     FORMAT(4D15.6)
00560            GO TO 1
00570    2       CONTINUE
00580            XNEW=XNEW-XINCR
00590            DX=-AL*(XJ+BE)
```

```
00600          CALL EXTREM(DX,AZ2,AZP2,BZ2,BZP2,Z2)
00610          DEX=DEXP(Z-Z2)
00620          C11=PI*((AZ2*BZP*DEX)-((BZ2*AZP)/DEX))
00630          C12=(PI*TT)*(((BZ2*AZ)/DEX)-(AZ2*BZ*DEX))
00640          C21=(PI/TT)*((AZP2*BZP*DEX)-((BZP2*AZP)/DEX))
00650          C22=PI*(((BZP2*AZ)/DEX)-(AZP2*BZ*DEX))
00660          UJ=C11*U+C12*UP
00670          UPJ=C21*U+C22*UP
00680          UPJ=UPJ*TT
00690          IF(MFLAG.NE.1) RETURN
00700          SEND=(UINIT**2)+(UJ**2)
00710          ZNORM=(XINCR/3.0D0)*(SUME+SUMO+SEND)
00720          XNORM=XNORM+ZNORM
00730          RETURN
00740          END
```

```
      SUBROUTINE LG(AL,BE,X,XJ,U,UJ,UP,UPJ,XNEW,XINCR,TT,
     1 X1SQ,X2SQ)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /IFLAGS/ MFLAG,MFLAG1,NTYPE
      COMMON /JC/ JCOUNT
      DIMENSION W(250),WP(250)
      DM=11.0D0
      X1SQ=0.0D0
      X2SQ=0.0D0
      IA=0
      UINIT=U
      SUMO=0.0D0
      SUME=0.0D0
      PI=3.14159265358979323840D0
      DX=-AL*(X+BE)
      CALL EXTREM(DX,AZ,AZP,BZ,BZP,Z)
    1 XNEW=XNEW-XINCR
      FCT=1.0D0
      IF(XNEW.LT.XJ) GO TO 2
      DX=-AL*(XNEW+BE)
      IF(DX.GT.DM.OR.IA.LT.1) GO TO 4
      XNEW=XNEW+XINCR
      IF(MFLAG.NE.1) GO TO 5
      IM=IA-1
      IF(IM.LT.1) GO TO 20
      DO 14  K=1,IM,2
   14 IF(DABS(W(K)).GT.1.D-18) SUMO=SUMO+(2.0D0*W(K)*W(K))
      IF(IM.LT.2) GO TO 20
      DO 15 K=2,IM,2
   15 IF(DABS(W(K)).GT.1.D-18) SUME=SUME+(4.0D0*W(K)*W(K))
   20 CONTINUE
      SEND=(UINIT**2)*W(IA)**2
      X1SQ=(XINCR/3.0D0)*(SUME+SUMO+SEND)
  100 FORMAT(5D14.3)
    5 CONTINUE
      CALL PR(AL,BE,XNEW,XJ,U,UJ,UP,UPJ,XNEW,XINCR,TT,
     1 X2SQ,Y2SQ)
      RETURN
    4 CONTINUE
      CALL EXTREM(DX,AZ2,AZP2,BZ2,BZP2,Z2)
      DEX=DEXP(Z-Z2)
      C11=PI*((AZ2*BZP*DEX)-((BZ2*AZP)/DEX))
      C12=(PI*TT)*(((BZ2*AZ)/DEX)-(AZ2*BZ*DEX))
      C21=(PI/TT)*((AZP2*BZP*DEX)-((BZP2*AZP)/DEX))
      C22=PI*(((BZP2*AZ)/DEX)-(AZP2*BZ*DEX))
      U1=C11*U+C12*UP
   11 FORMAT(1D20.8)
      U1P=(C21*U+C22*UP)*TT
      TT=1.0D0
      IA=IA+1
  502 FORMAT(I3)
      W(IA)=U1
      WP(IA)=U1P
      U=W(IA)
      UP=WP(IA)
   30 FORMAT(I3,D)
      IF(MFLAG1.EQ.1) WRITE(NTYPE,11) W(IA)
      DX=-AL*(XNEW+BE)
      AZ=AZ2
      AZP=AZP2
```

```
00670            BZ=BZ2
00680            BZP=BZP2
00690            Z=Z2
00700            GO TO 1
00710    2       CONTINUE
00720            IF(MFLAG.NE.1) GO TO 3
00725            IF(IA.LT.1) GO TO 3
00730            DO 16 K=1,IA,2
00740    16      IF(DABS(W(K)).GT.1.D-18) SUMO=SUMO+2.0D0*W(K)**2
00745            IF(IA.LT.2) GO TO 3
00750            DO 17 K=2,IA,2
00760    17      IF(DABS(W(K)).GT.1.D-18) SUME=SUME+4.0D0*W(K)**2
00770    3       CONTINUE
00780            XNEW=XNEW+XINCR
00790            DX=-AL*(XJ+BE)
00795            IF(DX.LT.5) TYPE 28,DX
00798    28      FORMAT('   WARNING: IN LG ARG OF EXTREM IS ',D)
00800            CALL EXTREM(DX,AZ2,AZP2,BZ2,BZP2,Z2)
00810            DEX=DEXP(Z-Z2)
00820            C11=PI*((AZ2*BZP*DEX)-((BZ2*AZP)/DEX))
00830            C12=(PI*TT)*(((BZ2*AZ)/DEX)-(AZ2*BZ*DEX))
00840            C21=(PI/TT)*((AZP2*BZP*DEX)-((BZP2*AZP)/DEX))
00850            C22=PI*(((BZP2*AZ)/DEX)-(AZP2*BZ*DEX))
00860            UJ=C11*U+C12*UP
00870            UPJ=(C21*U+C22*UP)*TT
00880            IF(MFLAG.NE.1) RETURN
00930            SEND=(UINIT**2)+(UJ**2)
00940            X1SQ=(XINCR/3.0D0)*(SUME+SUMO+SEND)
00950            RETURN
00960            END
```